10-2022

VPSL: Verifiable privacy-preserving data search for cloud-assisted Internet of Things

Qiuyun TONG
Xidian University

Yinbin MIAO
Xidian University

Ximeng LIU
Fuzhou University

Kim-Kwang Raymond CHOO
University of Texas at San Antonio

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

VPSL: Verifiable Privacy-Preserving Data Search for Cloud-Assisted Internet of Things

Qiuyun Tong, Yinbin Miao, Ximeng Liu, Kim-Kwang Raymond Choo, *Senior Member, IEEE,* Robert H. Deng, *Fellow, IEEE,* and Hongwei Li

***Abstract*—Cloud-assisted Internet of Things (IoT) is increasingly prevalent used in various fields, such as the healthcare system. While in such a scenario, sensitive data (e.g., personal electronic medical records) can be easily revealed, which incurs potential security challenges. Thus, Symmetric Searchable Encryption (SSE) has been extensively studied due to its capability of supporting efficient search on encrypted data. However, most SSE schemes require the data owner to share the complete key with query users and take malicious cloud servers out of consideration. Seeking to address these limitations, in this paper we propose a Verifiable Privacy-preserving data Search scheme with Limited key-disclosure (VPSL) for cloud-assisted Internet of Things. VPSL first designs a trapdoor generation protocol for obtaining a trapdoor with disclosing limited key information and without revealing plaintext query points to others. Then, VPSL provides an efficient result verification and search processing by employing the Merkle hash tree structure and k-means clustering technique, respectively. VPSL is secure against the level-2 attack. Finally, an enhanced VPSL (called VPSL+) resisting the level-3 attack is constructed by introducing the random splitting technique. Empirical experiments demonstrate the accuracy and efficiency of VPSL or VPSL+ using real-world datasets.**

***Index Terms*—Cloud-assisted Internet of Things, Searchable symmetric encryption, k-means clustering technique, Result verification.**

I. Introduction

INTERNET of Things (IoT), formed by combining various information sensing devices (i.e., sensor, smartphone, etc.) with the Internet to realize information gather, management and computation, has been wildly applied in a range of scenarios such as smart grid, smart home, smart healthcare system, and smart vehicle network [1], [2], [3], [4]. Due to a great amount of IoT data (e.g., Electronic Medical Records (EMRs), financial documents) produced and the limitation of IoT devices (i.e., limited storage and computation capacities, etc.), cloud-assisted IoT has been proposed to release storage and computation overheads and improve flexibility and convenience. Although the attractive benefits brought by cloud-assisted IoT, security and privacy risks are still challenging concerns. Fine-grained access control over the encrypted data [5] can achieve secure data sharing, but it may incur the waste of storage and computation resources due to returning all matched results. Thus, Searchable Encryption (SE) [6], [7] has been extensively studied, which allows users to search on ciphertexts. The scheme [8] proves it practicality and feasibility.

Q. Tong and Y. Miao (corresponding author) are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China; State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China. E-mail: ybmiao@xidian.edu.cn.

X. Liu is with the Key Laboratory of Information Security of Network Systems, School of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China. Email: snbnix@gmail.com.

K.-K. R. Choo is with the School of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA. Email: raymond.choo@fulbrightmail.org.

R. H. Deng is with the School of Information Systems, Singapore Management University, 80 Stamford Road 178902, Singapore. Email: robertdeng@smu.edu.sg.

H. Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China. Email: hongweili@uestc.edu.cn.

A. Limitations

Traditional SE solutions focus on a broad range of search functionalities such as single keyword search [6], [9], multi-keyword boolean search [10], and multi-keyword ranked search [11], [12], [13], [14], [15], [16]. To avoid returning large number of irrelevant results or incurring poor search experience, multi-keyword ranked search is generally more popular. Unfortunately, there are still some limitations in existing multi-keyword ranked search schemes based on secure k Nearest Neighbor (kNN) technique [17]. Examples include heavy search burden for tons of data, complete key disclosure and false results– see Fig. 1, which limit their practical applications.

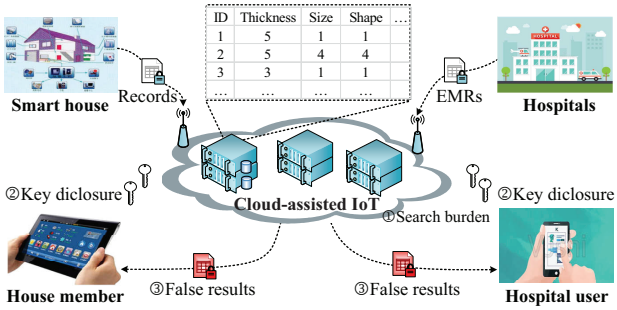


Fig. 1: Example limitations in existing schemes.

The search complexity in [11], [12] grows linearly with the size of document set. Existing solutions, such as those presented in [13], [14], achieve sublinear search complexity without damaging search accuracy. The schemes [15], [16] support more efficient retrieval at the expense of some search accuracy. However, these solutions require the data owner to share the complete secret key with authorized query users, which increases key leakage risk because interest-driven query users may leak the secret key to adversaries (e.g., unauthorized

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

query users) [18]. In addition, most above schemes except [15] assume the clouds to be honest-but-curious, who honestly follow the established protocol but are sufficiently curious to try to infer some valuable information from the encrypted data. In practice, this assumption seems to be insufficient due to financial incentives, hacker attacks or hardware errors, etc. [19], [20], [21], [22], [23], [24]. As a result, the clouds may just execute partial search operations. Thus, we hope to have a more trustworthy secure search system, where the authentic search results are assured to be returned despite the existence of malicious clouds. Accumulator structure, homomorphic MAC and Merkle hash tree have been used to prevent malicious clouds from returning false results. Accumulator structure [19], [20] incurs unbearable computation overhead due to exponential operation. Although homomorphic MAC [21] achieves efficient verification, it leads to a heavy index storage burden and high proof generation overhead. In addition, it just supports private verification (i.e., specific query users with the secret key or other information are allowed to verify search results) [22], [23], which incurs additional storage and computation costs, particularly impractical for resource-constrained IoT devices/users (mobile devices, sensor nodes, etc.). Merkle hash tree [15], [24] achieves efficient public result verification by returning auxiliary information.

B. Our Contributions

In this paper, we first design a Verifiable Privacy-preserving data Search scheme with Limited key disclosure (VPSL) for cloud-assisted IoT, by using k-means clustering technique, Asymmetric Scalar-Product-preserving Encryption technique (ASPE) and Merkel hash tree to achieve efficient ciphertext retrieval, limited key disclosure and efficient result verification simultaneously, which resists the level-2 attack[1]. Then, to provide stronger security, we construct an enhanced VPSL (called VPSL+) by adding the random splitting technique to index and trapdoor encryption, which resists the level-3 attack[2]. The main constructions of this paper are listed below:

- *Efficient index tree structure*. We use the k-means clustering technique iteratively to divide the data record set into multiple clusters until there is no cluster whose size is more than the threshold T, Then, we build an index tree by taking the upper layer's clusters as the parents of the next layer's and the data records in each lowest cluster as this cluster's children to improve search efficiency.
- *Limited key disclosure*. We design a trapdoor generation protocol based on ASPE. By implementing this protocol cooperatively at the cost of a round of communication, the trapdoor can be generated without leaking complete secret key or revealing plaintext query. VPSL or VPSL+ not only achieves key confidentiality to some extent, but also reduces key management burden.
- *Efficient public result verification*. We efficiently verify the correctness of search results by using the Merkle hash tree structure and digital signature to prevent malicious clouds from forging or tempering search results. Moreover, VPSL or VPSL+ allows public result verification, namely any trusted authority can verify the search results, which improves the reliability of the cloud-assisted IoT.
- *Secure against the level-3 attack*. We improve the index and query encryption by integrating the random splitting technique into VPSL to enhance data confidentiality. Thus, VPSL+ can securely resist the level-3 attack.

In Sections II and III, we review the existing literature and relevant background knowledge. Section IV describes the system model, threat model and design goals. In Section V, we present the overview of VPSL or VPSL+ and corresponding concrete constructions. Security and performance evaluations of our proposed schemes are presented in Section VI and VII. Finally, we conclude this paper and discuss its future work in Section VIII.

II. Related Work

To facilitate search on encrypted data and prevent malicious clouds from misbehaving during the retrieval process, we will mainly focus on the following topics such as multi-keyword ranked search and result verification.

Multi-keyword ranked search. A large variety of SSE schemes with different functionalities have been presented [25], [26], [27], such as those providing multi-keyword ranked search [11], [12], [13], [14], [15], [21]. These SSE schemes achieved more efficient retrieval based on the secure kNN technique [17] in comparison to Asymmetric SE (ASE) schemes [10], [28]. But the search complexity of these schemes [11], [12], [21] is linear with the size of dataset. To achieve sublinear retrieval, Chen *et al.* [15] used the hierarchical clustering method to support more search semantics and fast ciphertext retrieval over large-scale datasets at the expense of some search accuracy. Xia *et al.* [14] constructed a special keyword balanced binary tree as the index and performed the greedy depth-first search on it to achieve multi-keyword ranked search without accuracy loss, but it is less efficient than [15]. However, these schemes are known to have potential key leakage risks as query users can access all keys generated by the data owner. Interest-driven query users may disclose the secret key to adversaries such as unauthorized users, thereby threatening the security of outsourced data. Seeking to minimize the risk of secret key disclosure, the data owner in [18] cooperated with each query user to generate corresponding trapdoor without sharing complete secret key with query users. Although this protocol is more efficient and feasible than other secure computation protocols such as garbled-circuit protocol [29], secret sharing [30] and homomorphic encryption system [31], [18] cannot guarantee the correctness of search results. Besides, its search complexity is linear with the size of dataset and higher than that of [11], [12].

Result verification. As we all known, a large number of privacy-preserving keyword researches have been presented to ensure search result authenticity against malicious clouds

[1]Level-2 attack is also called the Known-Sample Attack (KSA). In this attack model, the attacker observes the encrypted dataset and a set of plaintext tuples in the plaintext dataset, but he/she does not know the corresponding encrypted values of those tuples in the encrypted dataset.

[2]Level-3 attack is called the Known-Plaintext Attack (KPA). In this attack model, the attacker not only knows the encrypted dataset, but also observes a set of tuples and the corresponding encrypted values of those tuples.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

(e.g., [15], [19], [20], [21], [22], [32], [33], [34]). Specifically, schemes such as [21], [22] support private verification, where only the verifier with specific information (e.g., secret key) can check the correctness of search results. For example, Wan *et al.* [21] integrated the homomorphic MAC into the commonly used multi-keyword ranked search scheme [11] to support verifiable multi-keyword search, which achieves private verification and causes heavy index storage burden and high proof generation overhead. To realize public verification, Liu *et al.* [19] dynamically verified the correctness of search results using RSA accumulator at the expense of huge computation overhead. Sun *et al.* [20] constructed an authenticated inverted index tree based on the bilinear-map accumulator technique to verify results' correctness and completeness either privately by query users or with the assistance of a public trusted authority, but its verification cost grows linearly with the number of queried keywords. Chen *et al.* [15] extended the Merkle hash tree [35] to searchable index tree to achieve efficient public result verification, where the search time is sublinear growth with the size of dataset and the verification is more efficient than accumulator structure. In addition, verifiable dynamic symmetric searchable encryption has been achieved by the schemes [32], [33], [34], but they either support single keyword matching search, or enable result verification with two round communications in the single-user setting. TABLE I presents a comparative summary of previous works and our proposed schemes.

TABLE I: A comparative summary of various works

Schemes	$\mathbb{F}_1$	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_4$	$\mathbb{F}_5$
[11]	Multiple	✗	✗	✗	KPA
[13]	Multiple	✓	✗	Public	KPA
[14]	Multiple	✓	✗	✗	KPA
[15]	Multiple	✓	✗	Public	KPA
[17]	Multiple	✗	✗	✗	KSA/KPA
[18]	Multiple	✗	✓	✗	KSA
[19]	Single	✓	✗	Public	IND-CKA2
[20]	Multiple	✗	✗	Public/Private	UC
[21]	Multiple	✗	✗	Private	KPA
[22]	Single	✗	✗	Private	CKA
[33]	Single	✓	✗	Private	—
[34]	Single	✗	✗	Public	Forward security
VPSL	Multiple	✓	✓	Public	KSA
VPSL+	Multiple	✓	✓	Public	KPA

— $\mathbb{F}_1$: Keyword-based search type; $\mathbb{F}_2$: Sublinear query; $\mathbb{F}_3$: Limited key disclosure; $\mathbb{F}_4$: Result verification; $\mathbb{F}_5$: Resisted attack type.
— KSA: Known-sample attack; KPA: Known-plaintext attack; CKA: Chosen keyword attack; IND-CKA2: Adaptive semantic security against CKA; UC: Universally composable security.

III. Preliminaries

In this section, we review some background knowledge in our work, including ASPE [17] and Merkle hash tree [35].

A. ASPE

ASPE is a SSE technique used to encrypt two vectors and compute their Euclidean distance. To securely compute the Euclidean distance between the data vector $\mathbf{p}$ and query vector $\mathbf{q}$, ASPE first generates the secret key $\mathbf{M}$, where $\mathbf{M}$ is an invertible matrix utilized to encrypt these two vectors. Then, ASPE extends $\mathbf{p}, \mathbf{q}$ to $\hat{\mathbf{p}} = (\mathbf{p}^\top, -0.5||\mathbf{p}||^2)^\top, \hat{\mathbf{q}} = r(\mathbf{q}^\top, 1)^\top$, where $r > 0$ is a random number. After that, ASPE encrypts $\hat{\mathbf{p}}, \hat{\mathbf{q}}$ as $\mathbf{p}^* = \mathbf{M}^\top \hat{\mathbf{p}}, \mathbf{q}^* = \mathbf{M}^{-1}\hat{\mathbf{q}}$ respectively, and computes their inner product, which is equivalent to the Euclidean distance of $\mathbf{p}$ and $\mathbf{q}$. ASPE can attack the level-2 attack. To provide stronger security, enhanced ASPE (i.e., secure kNN technique) is provided, which uses two ASPEs with different secret keys to encrypt data vectors or query vectors. The specific algorithms of enhanced ASPE can be referred to [17].

B. Merkle Hash Tree

Merkle Hash Tree (MHT) is commonly used to verify whether a set of messages are unaltered or undamaged. The structure of MHT is often a binary tree built in a bottom-up manner, where the leaf nodes store the digests of authentic data and non-leaf nodes store the digests derived from their children. Fig. 2 shows an example of the binary MHT, where $h_i = H(f_i)(i \in [1,8]), h_{1,2} = H(h_1|h_2), h_{1-4} = H(h_{1,2}|h_{3,4}), h_r = H(h_{1-4}|h_{5-8})$ and $H(\cdot)$ is a one-way collision-resistance hash function such as SHA256. The verifier with root digest h_r verifies the correctness of returned results following two steps. First, the verifier recalculates root value h'_r with the returned results and auxiliary information defined as siblings' digests of the nodes on the search path. Then, the verifier compares h'_r with h_r. For example, to verify the messages $\{f_1, f_4, f_7\}$, the prover returns them along with the auxiliary information $\{h_2, h_3, h_{5,6}, h_8\}$. Then, the verifier checks the correctness of $\{f_1, f_4, f_7\}$ by computing and checking Eq. 1 holds or not. If Eq. 1 holds, it implies that the returned results $\{f_1, f_4, f_7\}$ are not unaltered or undamaged.

$$H((H(f_1)|h_2)|(h_3|H(f_4))|h_{5,6}|(H(f_7)|h_8)) \stackrel{?}{=} h_r. \quad (1)$$

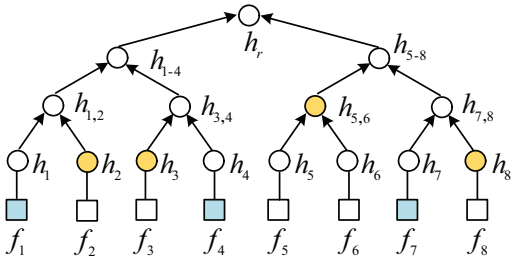


Fig. 2: An example of MHT.

IV. Problem Formulation

In this section, we present the system model[3], threat model and design goals of our proposed schemes.

A. System Model

VPSL or VPSL+ considers a cloud-assisted hospital IoT scenario, which mainly consists of three entities: Data Owner

[3]The system model of VPSL or VPSL+ may include an extra entity (verifier) apart from the data owner, the query users and the cloud server. Moreover, as the verifier can be any trusted authority, we omit it in Fig. 3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

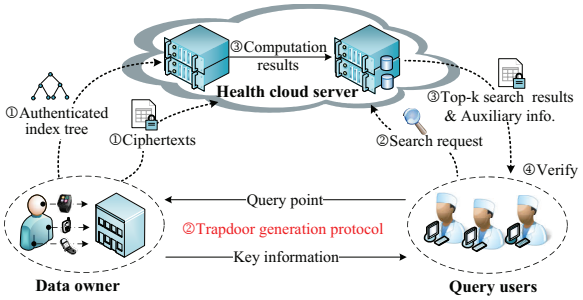


Fig. 3: System model of VPSL or VPSL+.

(DO), Query Users (QUs) and health Cloud Server (CS) as shown in Fig. 3. Before outsourcing EMRs to CS, DO (e.g., a hospital collects diagnostic information from patients via their IoT devices, and then records it in patients' EMRs) encrypts EMRs via a symmetric encryption algorithm (e.g., AES) and constructs an authenticated index tree to facilitate result verification and efficient retrieval (Step ①). If certain QU (e.g., a doctor with smart terminals) intends to know the risk factor of a certain disease (e.g., heart disease, breast cancer) in a specific patient based on the outsourced EMRs and the patient's test result, he/she cooperates with DO to perform a trapdoor generation protocol to obtain the trapdoor without knowing complete secret key (Step ②). Specifically, QU first sends his/her query (i.e., test result) to DO who then returns the secret key information generated by some confidential arithmetic operations. With the help of the key information, QU generates the trapdoor and sends it to CS. After receiving the search request, CS performs a search on the authenticated index tree, then finds top-k search results, finally returns them with auxiliary information (Step ③). Once obtaining the search results, the verifier (any trusted authority such as authorized QU) checks their correctness according to the auxiliary information (Step ④). The specific role of each entity in VPSL or VPSL+ is introduced as follows:

- *Data owner*. DO is responsible for generating the secret key required by the system, encrypting outsourced data record set, constructing an authenticated index tree and collaboratively executing the trapdoor generation protocol with QUs.
- *Query users*. QUs cooperate with DO to generate trapdoors, send them to CS, and verify the correctness of search results before decrypting them.
- *Health cloud server*. CS, having powerful storage and computation capabilities, stores huge amounts of data from DO and provides ciphertext retrieval services for authorized QUs. After receiving search requests, CS searches the authenticated index tree, and returns the search results attached with the auxiliary information for verification.

B. Threat Model

Different from the frequently-used threat model in which CS is assumed to be honest-but-curious, VPSL or VPSL+ considers a challenging one where CS is a malicious entity who may return tempered or forged results due to financial incentives, hacker attacks or hardware errors, etc. [19], [20], [21], [22], [23], [24]. Also, we consider that some corrupted QUs will collude with adversaries such as unauthorized QUs to release the secret key. Besides, DO is considered as an honest-but-curious entity who is curious about QUs' query contents. However, DO or QUs cannot collude with CS. It is remarkable that VPSL or VPSL+ focuses on the data confidentiality and correctness of search results, and takes no account of access pattern and search pattern in our discussion.

C. Design Goals

To achieve verifiable, secure and efficient encrypted data retrieval with limited key leakage under the above threat model, VPSL or VPSL+ should achieve the following goals regarding privacy preserving, efficiency and verifiability:

Privacy preserving. VPSL or VPSL+ should meet the privacy requirements:

- *Data confidentiality.* The outsourced encrypted dataset in VPSL or VPSL+ should be secure against the level-2 attack or level-3 attack, respectively; namely, the plaintext dataset will not be revealed according to the leaked information.
- *Query confidentiality.* VPSL or VPSL+ requires that each plaintext query should not be leaked to DO or CS throughout the encrypted data retrieval.
- *Trapdoor unlinkability.* To prevent CS from deducing any useful information through analyzing the relationship between trapdoors, VPSL or VPSL+ should ensure that CS does not determine whether two different trapdoors are driven from the same query point.

Efficiency. VPSL or VPSL+ should achieve sublinear search time, and efficient result verification without imposing high computation overhead on resources-limited QUs.

Verifiability. VPSL or VPSL+ should prevent the malicious CS from returning false query results and thereby verifying the correctness of search results.

V. Proposed VPSL

In this section, we first give a high-level description of VPSL or VPSL+, then present concrete construction of VPSL to achieve efficient encrypted data retrieval, limited key disclosure and efficient result verification, finally construct VPSL+ concretely to provide stronger security at the expense of a little computation and communication costs.

A. Overview of VPSL & VPSL+

VPSL. To solve the limitations, namely heavy search burden, complete key leakage and false search results, DO first uses the k-means clustering technique iteratively until no cluster has more than T items, then builds a plaintext index tree based on the clustering result to improve search efficiency. Next, VPSL constructs a Trapdoor Generation Protocol (TGP) based on the protocol in scheme [18] to encrypt a query point with a round of communication between DO and QU. At the expense of DO online and a small amount of communication

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

cost, QUs' query information can be protected from DO and CS and trapdoor generation with limited key disclosure can be achieved. Although, ASE schemes can also resolve the threat of key disclosure, they cause heavy computation burden comparing with TGP. Finally, VPSL authenticates the encrypted index tree base on MHT to support efficient public result correctness verification. Compared with accumulator structure and homomorphic MAC technique, MHT used in our proposed schemes is more efficient to support public verification.

VPSL+. Although VPSL achieves efficient encrypted data search, limited key disclosure and efficient result verification, it is only secure against the level-2 attack [17]. Therefore, we improve VPSL to resist the level-3 attack in VPSL+. To address this issue, we integrate the random splitting technique into the index and query encryption.

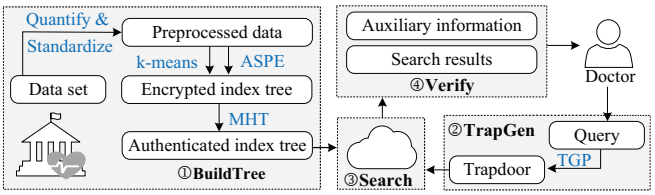


Fig. 4: Overview of VPSL or VPSL+.

The overview of VPSL or VPSL+, composed of four algorithms except **KeyGen**, can be seen in Fig. 4. In Step ①, DO first transforms string information into numeric data (i.e., qualify dataset), then standardizes each attribute of the transformed data (e.g., using z-score standardization) to prevent the k-means clustering's effect from being affected by different dimensions and the large numerical difference between the attributes. For example, $p_1 = (2, 1000), p_2 = (4, 6000), p_3 = (3, 2000)$ are three data records with two attributes. Standardizing them via z-score standardization method, we obtain $p_1 = (-\frac{1}{\sqrt{2}}, -\frac{2}{\sqrt{41}}), p_2 = (\frac{1}{\sqrt{2}}, \frac{3}{\sqrt{41}}), p_3 = (0, -\frac{1}{\sqrt{41}})$. After that, the preprocessed data will be divided into several clusters with size not exceeding threshold T by using the k-means clustering technique iteratively. Finally, an encrypted index tree is constructed with the help of ASPE, which will be authenticated later based on MHT. In Step ②, DO and QU cooperate to implement TGP to encrypt plaintext query point without sharing complete secret key with QU or revealing plaintext query point to DO. In Step ③, CS retrieves the authenticated index tree from the root to find the best matching cluster, then computes the relevance scores between the query and data records contained in this cluster. If the size of this cluster cannot satisfy QU's requirement, CS will search its sibling cluster with the highest relevance score among other sibling clusters until meeting the requirement. In Step ④, the verifier such as an authorized QU verifies the correctness of search results by using the returned auxiliary information. As for the specific process in different algorithms in VPSL or VPSL+, we will give a briefly definition as follows.

- **KeyGen**$(1^\kappa) \rightarrow (\mathcal{K}, K)$. In this stage, DO first generates the secret key $\mathcal{K} = \{\alpha, \mathbf{r}, \mathbf{s}, \widehat{\mathbf{M}}^{-1}\}$ in VPSL, or $\mathcal{K} = \{\alpha, \mathbf{r}, \mathbf{s}, \mathbf{S}, \widehat{\mathbf{M}}^{-1}, \widetilde{\mathbf{M}}^{-1}\}$ in VPSL+, and then derives the secret key information $K = \{\mathbf{g}, \widehat{\mathbf{M}}^*, \widehat{\mathbf{M}}^\circ\}$ based on $\widehat{\mathbf{M}}$ in VPSL or $K = \{\dot{\mathbf{g}}, \ddot{\mathbf{g}}, \widehat{\mathbf{M}}^*, \widetilde{\mathbf{M}}^*, \widehat{\mathbf{M}}^\circ, \widetilde{\mathbf{M}}^\circ\}$ based on $\widehat{\mathbf{M}}, \widetilde{\mathbf{M}}$ in VPSL+ to prepare for trapdoor generation in TGP. Here, $\widehat{\mathbf{M}}, \widetilde{\mathbf{M}}$ are two random invertible matrices.
- **BuildTree**$(F, \mathcal{K}) \rightarrow \Gamma^*$. After preprocessing the data record set F, DO first constructs an encrypted index tree Γ' by using the k-means clustering technique and secret key $\mathcal{K}$, then authenticates it with MHT to obtain the authenticated index tree Γ^*.
- **TrapGen**$(\mathbf{q}, \mathbf{t}, K) \rightarrow \mathbf{T_q}$. When ceratin QU issues the query point $\mathbf{q}$, DO and QU perform TGP cooperatively with the help of a random vector $\mathbf{t}$ and the secret key information K to obtain the trapdoor $\mathbf{T_q}$ without sharing complete secret key with QU or revealing the plaintext query point $\mathbf{q}$ to DO.
- **Search**$(\Gamma^*, \mathbf{T_q}, k) \rightarrow (\mathcal{R}, VO)$. When receiving the query request $\{\mathbf{T_q}, k\}$, CS performs a search on the authenticated index tree Γ^* to find the matching clusters and select top-k similar encrypted data records from these clusters as search results $\mathcal{R}$, and returns them with the auxiliary information VO for verifiability.
- **Verify**$(VO, \mathcal{R}) \rightarrow True/False$. To verify the correctness of search results, the verifier recomputes the root digest and compares it with the real. If the two digests are equal, the verifier returns $True$ and returns $False$ otherwise.

B. Construction of VPSL

The concrete construction of VPSL consists of five algorithms described below.

KeyGen(1^κ). Given the security parameter κ, DO first randomly generates a positive number α, two $(n+1)$-dimensional vectors $\mathbf{r} = (r_1, \cdots, r_{n+1})$, $\mathbf{s} = (s_1, \cdots, s_{n+1})$, and a $(2n+2) \times (2n+2)$-dimensional invertible matrix $\widehat{\mathbf{M}}$ to obtain the secret key $\mathcal{K} = \{\alpha, \mathbf{r}, \mathbf{s}, \widehat{\mathbf{M}}^{-1}\}$. Then, DO computes $(2n+2)$-dimensional vector $\mathbf{g}$ and two $(2n+2) \times n$-dimensional matrices $\widehat{\mathbf{M}}^*, \widehat{\mathbf{M}}^\circ$ (see Eq. 2). Besides, DO generates a symmetric key sk to encrypt the data record set F to obtain ciphertexts, i.e., $\{c_i\} = \{Enc_{sk}(f_i)\}$.

$$\begin{aligned} &\mathbf{g} = (\cdots, \mathbf{M}_{i,2n+1} + \mu \mathbf{M}_{i,2n+2}, \cdots), \\ &\widehat{\mathbf{M}}_i^* = (\cdots, \widehat{\mathbf{M}}_{i,2j} - \widehat{\mathbf{M}}_{i,2j-1}, \cdots), \\ &\widehat{\mathbf{M}}_i^\circ = (\cdots, \widehat{\mathbf{M}}_{i,2j-1}, \cdots), \\ &i \in [1, 2n+2], j \in [1, n]. \end{aligned} \tag{2}$$

Here, n is the number of attributes in the dataset F. μ is a random number. $\widehat{\mathbf{M}}_{i,j}$ is the i-th row, j-th column of $\widehat{\mathbf{M}}$. $\widehat{\mathbf{M}}_i^*, \widehat{\mathbf{M}}_i^\circ$ are the i-row of $\widehat{\mathbf{M}}^*, \widehat{\mathbf{M}}^\circ$, respectively. The function of tuple $K = \{\mathbf{g}, \widehat{\mathbf{M}}^*, \widehat{\mathbf{M}}^\circ\}$ is help to generate the trapdoor with limited key leakage (detail in **TrapGen**).

BuildTree$(F, \mathcal{K})$. Given the dataset $F = \{f_1, \cdots, f_m\}$ containing m data records with n attributes, DO first qualifies and standardizes F to obtain the preprocessed data $P = \{\mathbf{p}_1, \cdots, \mathbf{p}_m\}$, where $\mathbf{p}_i = (p_{i,1}, \cdots, p_{i,n})$ $(i \in [1, m])$ is a n-dimensional vector, then builds an authenticated index tree Γ^* with four steps.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

`Step 1`: DO divides these data vectors $\{\mathbf{p}_1, \cdots, \mathbf{p}_m\}$ into several clusters $\{\mathbf{v}_1, \cdots, \mathbf{v}_u\}$ by recursively using the k-means clustering technique [16] until the number of data records in each cluster is not more than the threshold T (see Fig. 5a), where u is the number of non-leaf nodes at the bottom of the index tree Γ and $u \ll m$. Note that cluster $\mathbf{v}_i (i \in [1, u])$ is also a n-dimensional vector, so we set $\mathbf{v}_i = (v_{i,1}, \cdots, v_{i,n})$.

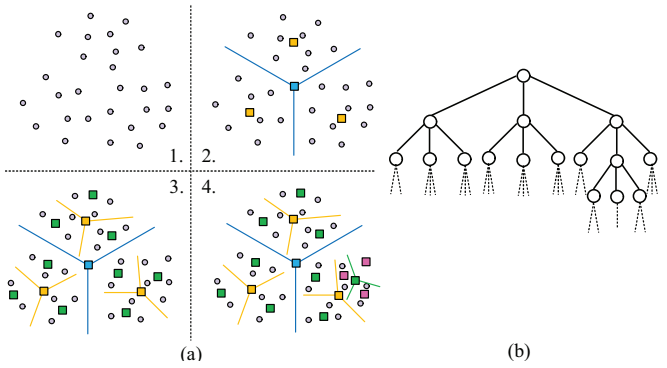


Fig. 5: Example of k-means clustering process and plaintext index tree construction with setting $T = 3, N = 3$, where $u = 11$ and N denotes the tree's width.

`Step 2`: Taking the upper layer's clusters as the parents of the next layer's, DO builds a plaintext index tree Γ (see Fig. 5b). Remarkably, all data vectors are stored in the leaf node level. Data vectors from the same cluster will connect to the same non-leaf node as their parent.

`Step 3`: DO encrypts the plaintext index tree Γ with the secret key $\mathcal{K}$ to obtain the encrypted index tree Γ'. Specifically, for each data vector $\mathbf{p}_i = (p_{i,1}, \cdots, p_{i,n}) (i \in [1, m])$, DO extends it to a $(2n+2)$-dimensional vector $\hat{\mathbf{p}}_i = (r_1 - 2\alpha p_{i,1}, s_1, \cdots, r_n - 2\alpha p_{i,n}, s_n, r_{n+1} + \alpha \sum_{j=1}^{n} p_{i,j}^2, s_{n+1})$, and encrypts $\hat{\mathbf{p}}_i$ as $\mathbf{p}_i^* = \hat{\mathbf{p}}_i \widehat{\mathbf{M}}^{-1}$. Similar to the data vector encryption, each cluster vector $\mathbf{v}_i = (v_{i,1}, \cdots, v_{i,n})$ is also extended and encrypted as $\mathbf{v}_i^* = \hat{\mathbf{v}}_i \widehat{\mathbf{M}}^{-1}$, where $\hat{\mathbf{v}}_i = (r_1 - 2\alpha v_{i,1}, s_1, \cdots, r_n - 2\alpha v_{i,n}, s_n, r_{n+1} + \alpha \sum_{j=1}^{n} v_{i,j}^2, s_{n+1})$. **Algorithm 1** describes the construction of the encrypted index tree Γ' for the data vector set P, where $root$ is the root node of Γ', P_i is a set formed by the data vectors in cluster $\mathbf{v}_i$ and m_i is the size of the set P_i. Note that the initial state of $root$ is an empty node.

`Step 4`: DO authenticates Γ' based on MHT to obtain the authenticated index tree Γ^*, whose each internal node consists of two components: encrypted cluster vector and digest, each leaf node stores one more component: ciphertext generated by encrypting corresponding data record, and root node stores one more component: signature. For each leaf node V_i, the digest is defined as $d_i = H(id_i | c_i | \mathbf{p}_i^*)$, where id_i and c_i denote the identity and ciphertext of the data record f_i, respectively. For each internal node V_ρ, assuming that V_ρ with encrypted cluster vector $\mathbf{v}_\rho^*$ has l children whose digests are denoted as $d_1, \cdots, d_l$, then the digest of V_ρ is $d_\rho = H(\mathbf{v}_\rho^* | H(d_1 | \cdots | d_l))$. For the root node V_r with T children, assuming that the digests of its children are denoted as $d_1, \cdots, d_T$, then the digest of V_r is $d_r = H(\mathbf{v}_r^* | H(d_1 | \cdots | d_T))$. Finally, DO signs the root digest d_r to obtain the signature S_{root} by using a digital signature technique (e.g., RSA signature). Here, $H(\cdot)$ is a one-way collision-resistance hash function (e.g., SHA256).

Algorithm 1: BuildEree$(P, T, N, root)$

```
{v_i, P_i}_{i∈[1,N]} = k-means(P, N); /* divide P into N clusters and assume P_i = {p_1, ..., p_{m_i}} */
for i = 1; i ≤ N; i++ do
    v_i^* = Encrypt(v_i, K);
    root.child[i] = v_i^*;
    if m_i ≤ T then
        for j = 1; j ≤ m_i; j++ do
            p_j^* = Encrypt(p_j, K);
            root.child[i].child[j] = p_j^*;
    else
        BuildEtree(P_i, T, N, root.child[i]);
```

TrapGen$(\mathbf{q}, \mathbf{t}, K)$. To generate a trapdoor $\mathbf{T_q}$ for the pre-processed query point $\mathbf{q} = (q_1, \cdots, q_n)$ with limited key disclosure, QU and DO cooperate to perform TGP at the expense of a round of communication cost. It is worthwhile to note that the communications between QUs and DO rely on a secure channel such as Secure Sockets Layer (SSL). The specific TGP is described as follows:

`Step 1`: QU randomly selects a n-dimensional vector $\mathbf{t} = (t_1, \cdots, t_n)$, performs vector addition operation, i.e., $\hat{\mathbf{q}} = \mathbf{q} + \mathbf{t} = (q_1 + t_1, \cdots, q_n + t_n)$, and sends $\hat{\mathbf{q}}$ to DO. Note that the function of $\mathbf{t}$ is to hide the real query point in order to protect each QU's query privacy from DO.

`Step 2`: DO randomly selects a positive number $\beta_\mathbf{q}$, computes a $(2n+2)$-dimensional vector $\mathbf{x} = \beta_\mathbf{q}(\mathbf{g} + \hat{\mathbf{q}}(\widehat{\mathbf{M}}^\circ)^\top)$ and a $(2n+2) \times n$-dimensional matrix $\mathbf{Y} = \beta_\mathbf{q} \widehat{\mathbf{M}}^*$, and sends $\mathbf{x}, \mathbf{Y}$ to QU.

`Step 3`: QU randomly selects a positive number $\gamma_\mathbf{q}$. Then, the trapdoor is

$$\mathbf{T_q} = \gamma_\mathbf{q}(\mathbf{x} + \mathbf{t}\mathbf{Y}^\top) = \beta_\mathbf{q}\gamma_\mathbf{q}\widehat{\mathbf{M}}(\bar{\mathbf{q}})^\top. \tag{3}$$

where $\bar{\mathbf{q}} = (q_1, t_1, \cdots, q_n, t_n, 1, \mu)$.

Search$(\Gamma^*, \mathbf{T_q}, k)$. After gaining the search request $\{\mathbf{T_q}, k\}$, CS retrieves the authenticated index tree Γ^* from the root node to find the cluster with highest similarity to the query $\mathbf{q}$, note that we use Euclidean distance as similarity measure metric. Specifically, at each level, CS finds the entry node of the next level, which has the highest relevance score (i.e., the smallest Euclidean distance). Up to the leaf node, CS computes and ranks the relevance scores of the data records contained in this cluster. If the size of the best matching cluster cannot satisfy QU's requirement, CS will search its sibling cluster with the second highest relevance score until the number of desired data records is satisfied. Finally, CS returns top-k search results $\mathcal{R}$ and the auxiliary information VO including all components (except the digest) of each node on the search path and the digests of its sibling nodes.

The relevance score between encrypted data vector $\mathbf{p}_i^*$ and trapdoor $\mathbf{T_q}$ is computed in Eq. 4. Then, CS can achieve relevance comparison between different data records according

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

to $\mathbf{p}_i^* \cdot \mathbf{T_q} - \mathbf{p}_j^* \cdot \mathbf{T_q} = \alpha\beta_\mathbf{q}\gamma_\mathbf{q}((||\mathbf{p}_i||^2 - 2\mathbf{p}_i\mathbf{q}^\top) - (||\mathbf{p}_j||^2 - 2\mathbf{p}_j\mathbf{q}^\top)) = \alpha\beta_\mathbf{q}\gamma_\mathbf{q}(D(\mathbf{p}_i,\mathbf{q}) - D(\mathbf{p}_j,\mathbf{q}))$, where $D(\cdot)$ donates Euclidean distance function. If $\mathbf{p}_i^* \cdot \mathbf{T_q} > \mathbf{p}_j^* \cdot \mathbf{T_q}$, then $D(\mathbf{p}_i,\mathbf{q}) > D(\mathbf{p}_j,\mathbf{q})$ because of $\alpha, \beta_\mathbf{q}, \gamma_\mathbf{q} > 0$, which indicates that the data vector $\mathbf{p}_j$ is more similar with query point $\mathbf{q}$ than the data vector $\mathbf{p}_i$. Note that the relevance scores between cluster vectors and query point are calculated and compared in similar way.

$$\begin{aligned}\mathbf{p}_i^* \cdot \mathbf{T_q} &= \hat{\mathbf{p}}_i\widehat{\mathbf{M}}^{-1} \cdot \beta_\mathbf{q}\gamma_\mathbf{q}\widehat{\mathbf{M}}\bar{\mathbf{q}}^\top \\ &= \alpha\beta_\mathbf{q}\gamma_\mathbf{q}(||\mathbf{p}_i||^2 - 2\mathbf{p}_i\mathbf{q}^\top) + \beta_\mathbf{q}\gamma_\mathbf{q}\sum_{j=1}^{n}(r_j q_j \\ &+ s_j t_j) + \beta_\mathbf{q}\gamma_\mathbf{q}(\mu s_{n+1} + r_{n+1}).\end{aligned} \tag{4}$$

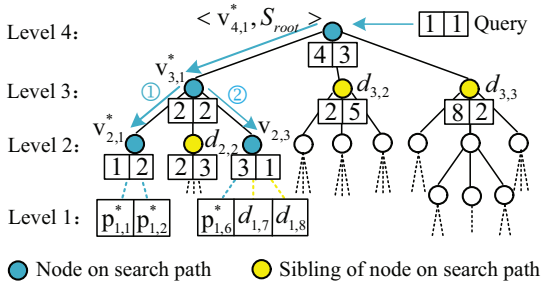


Fig. 6: Example of top-3 search with assuming $n = 2$.

Example. The above shows that encryption has no influence on data records' order ranked by their relevance scores. Thus, we will perform a search on the plaintext index tree with plaintext query in Fig. 6 to demonstrate the search process. Given the search query $\mathbf{q} = (1, 1)$ and $k = 3$, CS computes the relevance scores from the top of Γ. As $D(\mathbf{v}_{3,1},\mathbf{q}) = \sqrt{2} < D(\mathbf{v}_{3,2},\mathbf{q}) = \sqrt{17} < D(\mathbf{v}_{3,3},\mathbf{q}) = \sqrt{50}$, CS chooses $V_{3,1}$ as entry node and computes $D(\mathbf{v}_{2,1},\mathbf{q}) = 1, D(\mathbf{v}_{2,2},\mathbf{q}) = \sqrt{5}, D(\mathbf{v}_{2,3},\mathbf{q}) = 2$. Since $D(\mathbf{v}_{2,1},\mathbf{q}) < D(\mathbf{v}_{2,3},\mathbf{q}) < D(\mathbf{v}_{2,2},\mathbf{q})$, CS calculates $D(\mathbf{p}_{1,1},\mathbf{q}), D(\mathbf{p}_{1,2},\mathbf{q})$. As $V_{2,1}$ is the lowest non-leaf node and only has two children, which is less than $k = 3$, CS also calculates $D(\mathbf{p}_{1,6},\mathbf{q}), D(\mathbf{p}_{1,7},\mathbf{q}), D(\mathbf{p}_{1,8},\mathbf{q})$. Assuming $D(\mathbf{p}_{1,1},\mathbf{q}) < D(\mathbf{p}_{1,2},\mathbf{q}) < D(\mathbf{p}_{1,6},\mathbf{q}) < D(\mathbf{p}_{1,7},\mathbf{q}) < D(\mathbf{p}_{1,8},\mathbf{q})$, CS returns $\mathcal{R} = \{c_{1,1}, c_{1,2}, c_{1,6}\}$ as top-3 search results and the auxiliary information is $VO = \{\mathbf{p}_{1,1}^*, \mathbf{p}_{1,2}^*, \mathbf{p}_{1,6}^*, \mathbf{v}_{2,1}^*, \mathbf{v}_{2,3}^*, \mathbf{v}_{3,1}^*, d_{1,7}, d_{1,8}, d_{2,2}, d_{3,2}, d_{3,3}\}$.

Verify$(VO, \mathcal{R})$. To verify results' correctness, the verifier first uses $\mathcal{R}$ and VO to recalculate the root digest, then decrypts the public root signature S_{root} using the digital signature' public key to obtain the real root digest. Finally, the correctness of search results is verified by comparing these two digests. In our example, the verifier first recalculates the root digest $d'_{4,1}$ in Eq. 5, then decrypts the root signature S_{root} to obtain $d_{4,1}$. If $d'_{4,1}$ equals to $d_{4,1}$, then the search results are correct.

$$\begin{aligned} d'_{1,i} &= H(id_{1,i}|c_{1,i}|\mathbf{p}_{1,i}^*), i = 1, 2, 6, \\ d'_{2,1} &= H(\mathbf{v}_{2,1}^*|H(d'_{1,1}|d'_{1,2})), \\ d'_{2,3} &= H(\mathbf{v}_{2,3}^*|H(d'_{1,6}|d_{1,7}|d_{1,8})), \\ d'_{3,1} &= H(\mathbf{v}_{3,1}^*|H(d'_{2,1}|d_{2,2}|d'_{2,3})), \\ d'_{4,1} &= H(d'_{3,1}|d_{3,2}|d_{3,3}). \end{aligned} \tag{5}$$

C. Construction of VPSL+

VPSL+ continues to use the constructed plaintext index tree and result verification mechanism to achieve verifiable efficient multi-keyword search. To resist the level-3 attack, VPSL+ integrates the random splitting technique into index tree and query encryption, but we cannot simply integrate them. The reason is that the vector $\mathbf{t} = (t_1, \cdots, t_n)$ is split into two random vectors $\mathbf{t}^a, \mathbf{t}^b$ when DO splits the query vector $\hat{\mathbf{q}} = (q_1 + t_1, \cdots, q_n + t_n)$, but QU (without knowing $\mathbf{t}^a, \mathbf{t}^b$) can only use $\mathbf{t}$ and the knowledge returned by DO to generate the trapdoor, which will incur the relevance computation results including the item $t_\iota p_{i,\iota}$ when $\mathbf{S}[\iota] = 0$, where $\mathbf{S}$ is a random bit vector used in the random splitting technique. Thus, we need to improve the index or trapdoor extension to eliminate these items. As the construction of VPSL+ is based on that of VPSL, we just show the modified contents in **KeyGen**, **BuildTree**, **TrapGen**, **Search**.

KeyGen(1^κ). Given the security parameter κ, DO first invokes VPSL.KeyGen to obtain $\{\alpha, \mathbf{r}, \mathbf{s}, \widehat{\mathbf{M}}, \widehat{\mathbf{M}}^{-1}, sk\}$, then generates a $(n+1)$-dimensional bit vector $\mathbf{S}$ and a $(2n+2) \times (2n+2)$-dimensional invertible matrix $\widetilde{\mathbf{M}}$ to obtain the secret key $\mathcal{K} = \{\alpha, \mathbf{r}, \mathbf{s}, \mathbf{S}, \widehat{\mathbf{M}}^{-1}, \widetilde{\mathbf{M}}^{-1}\}$. Finally, DO computes two $(2n+2)$-dimensional vectors $\dot{\mathbf{g}}, \ddot{\mathbf{g}}$, and four $(2n+2) \times n$-dimensional matrices $\widehat{\mathbf{M}}^*, \widetilde{\mathbf{M}}^*, \widehat{\mathbf{M}}^\circ, \widetilde{\mathbf{M}}^\circ$ (see Eq. 6). Here, $\varepsilon_1, \varepsilon_2, \mu_1, \mu_2$ are different random numbers and $\varepsilon_1 + \varepsilon_2 = 1$ if $\mathbf{S}[n+1] = 0$; otherwise $\varepsilon_1 = \varepsilon_2 = 1$. Note that the dataset F in VPSL+ also encrypted via the symmetric key sk.

$$\begin{aligned} \dot{\mathbf{g}} &= (\cdots, \varepsilon_1\widehat{\mathbf{M}}_{i,2n+1} + \mu_1\widehat{\mathbf{M}}_{i,2n+2}, \cdots), \\ \ddot{\mathbf{g}} &= (\cdots, \varepsilon_2\widetilde{\mathbf{M}}_{i,2n+1} + \mu_2\widetilde{\mathbf{M}}_{i,2n+2}, \cdots), \\ \widehat{\mathbf{M}}_i^* &= (\cdots, \widehat{\mathbf{M}}_{i,2j} - \widehat{\mathbf{M}}_{i,2j-1}, \cdots), \\ \widetilde{\mathbf{M}}_i^* &= (\cdots, \widetilde{\mathbf{M}}_{i,2j} - \widetilde{\mathbf{M}}_{i,2j-1}, \cdots), \\ \widehat{\mathbf{M}}_i^\circ &= (\cdots, \widehat{\mathbf{M}}_{i,2j-1}, \cdots), \\ \widetilde{\mathbf{M}}_i^\circ &= (\cdots, \widetilde{\mathbf{M}}_{i,2j-1}, \cdots), \\ i &\in [1, 2n+2], j \in [1, n]. \end{aligned} \tag{6}$$

BuildTree$(F, \mathcal{K})$. The differences of authenticated index tree construction between VPSL and VPSL+ are reflected in `Step 3`: plaintext index tree encryption. Then index tree encryption in VPSL+ is shown as follows:

`Step 3'`: For each data vector $\mathbf{p}_i = (p_{i,1}, \cdots, p_{i,n})$ $(i \in [1, m])$, DO first extends it to a $(2n+2)$-dimensional vector $\hat{\mathbf{p}}_i = (r_1 - 2\alpha p_{i,1}, s_1 - (1 - \mathbf{S}[1]) \times 2\alpha p_{i,1}, \cdots, r_n - 2\alpha p_{i,n}, s_n - (1 - \mathbf{S}[n]) \times 2\alpha p_{i,n}, r_{n+1} + \alpha\sum_{j=1}^{n} p_{i,j}^2, s_{n+1})$, and encrypts $\hat{\mathbf{p}}_i$ as $\mathbf{p}_i^* = (\hat{\mathbf{p}}_i^a\widehat{\mathbf{M}}^{-1}, \hat{\mathbf{p}}_i^b\widetilde{\mathbf{M}}^{-1})$, where $\hat{\mathbf{p}}_i$ is split into two vectors $\hat{\mathbf{p}}_i^a, \hat{\mathbf{p}}_i^b$ by using bit vector $\mathbf{S}$ via Eq. 7, 8. Similar to the data vector encryption, each cluster vector $\mathbf{v}_i$ is also extended and encrypted as $\mathbf{v}_i^* = (\hat{\mathbf{v}}_i^a\widehat{\mathbf{M}}^{-1}, \hat{\mathbf{v}}_i^b\widetilde{\mathbf{M}}^{-1})$, where $\hat{\mathbf{v}}_i^a, \hat{\mathbf{v}}_i^b$ are obtained by splitting $\hat{\mathbf{v}}_i = (r_1 - 2\alpha v_{i,1}, s_1 - (1 - \mathbf{S}[1]) \times 2\alpha v_{i,1}, \cdots, r_n - 2\alpha v_{i,n}, s_n - (1 - \mathbf{S}[n]) \times 2\alpha v_{i,1}, r_{n+1} + \alpha\sum_{j=1}^{n} v_{i,j}^2, s_{n+1})$ via the same splitting configuration.

$$\begin{cases} \hat{\mathbf{p}}_i^a[2\iota - 1] = \hat{\mathbf{p}}_i^b[2\iota - 1] = \hat{\mathbf{p}}_i[2\iota - 1], if\ \mathbf{S}[\iota] = 0; \\ \hat{\mathbf{p}}_i^a[2\iota - 1] + \hat{\mathbf{p}}_i^b[2\iota - 1] = \hat{\mathbf{p}}_i[2\iota - 1], if\ \mathbf{S}[\iota] = 1; \end{cases} \tag{7}$$

$$\hat{\mathbf{p}}_i^a[2\iota] + \hat{\mathbf{p}}_i^b[2\iota] = \hat{\mathbf{p}}_i[2\iota], \iota \in [1, n+1]. \tag{8}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

TrapGen$(\mathbf{q}, \mathbf{t}, K)$. At the expense of a round of communication cost, the improved TGP is performed as follows to obtain the trapdoor $\mathbf{T_q}$ for the query point $\mathbf{q} = (q_1, \cdots, q_n)$ with limited key disclosure.

`Step 1′`: QU generates $\hat{\mathbf{q}} = \mathbf{q}+\mathbf{t} = (q_1+t_1, \cdots, q_n+t_n)$ with the same method in VPSL, then sends $\hat{\mathbf{q}}$ to DO.

`Step 2′`: DO first splits $\hat{\mathbf{q}}$ into two vectors $\hat{\mathbf{q}}^a, \hat{\mathbf{q}}^b$ (see Eq. 9), then randomly selects a positive number $\beta_{\mathbf{q}}$, next computes two $(2n+2)$-dimensional vectors $\dot{\mathbf{x}} = \beta_{\mathbf{q}}(\dot{\mathbf{g}} + \hat{\mathbf{q}}^a(\widehat{\mathbf{M}}^\circ)^\top), \ddot{\mathbf{x}} = \beta_{\mathbf{q}}(\ddot{\mathbf{g}} + \hat{\mathbf{q}}^b(\widetilde{\mathbf{M}}^\circ)^\top)$ and two $(2n+2) \times n$-dimensional matrices $\dot{\mathbf{Y}} = \beta_{\mathbf{q}}\widehat{\mathbf{M}}^*, \ddot{\mathbf{Y}} = \beta_{\mathbf{q}}\widetilde{\mathbf{M}}^*$, finally sends $\dot{\mathbf{x}}, \dot{\mathbf{Y}}, \ddot{\mathbf{x}}, \ddot{\mathbf{Y}}$ to QU.

$$\begin{cases} \hat{\mathbf{q}}^a[\iota] + \hat{\mathbf{q}}^b[\iota] = \hat{\mathbf{q}}[\iota], if\ \mathbf{S}[\iota] = 0; \\ \hat{\mathbf{q}}^a[\iota] = \hat{\mathbf{q}}^b[\iota] = \hat{\mathbf{q}}[\iota], if\ \mathbf{S}[\iota] = 1. \end{cases} \tag{9}$$

`Step 3′`: QU selects a random positive number $\gamma_{\mathbf{q}}$. Then, the trapdoor is obtained by Eq. 10.

$$\begin{aligned} \mathbf{T_q} &= (\mathbf{T}^a_{\mathbf{q}}, \mathbf{T}^b_{\mathbf{q}}) \\ &= (\gamma_{\mathbf{q}}(\dot{\mathbf{x}} + \mathbf{t}\dot{\mathbf{Y}}^\top), \gamma_{\mathbf{q}}(\ddot{\mathbf{x}} + \mathbf{t}\ddot{\mathbf{Y}}^\top)) \\ &= (\beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widehat{\mathbf{M}}(\bar{\mathbf{q}}^a)^\top, \beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widetilde{\mathbf{M}}(\bar{\mathbf{q}}^b)^\top). \end{aligned} \tag{10}$$

Here, $\bar{\mathbf{q}}^a = (\hat{\mathbf{q}}^a[1]-t_1, t_1, \cdots, \hat{\mathbf{q}}^a[n]-t_n, t_n, \varepsilon_1, \mu_1)$ and $\bar{\mathbf{q}}^b = (\hat{\mathbf{q}}^b[1] - t_1, t_1, \cdots, \hat{\mathbf{q}}^b[n] - t_n, t_n, \varepsilon_2, \mu_2)$.

Search$(\Gamma^*, \mathbf{T_q}, k)$. The search process is very similar to that in VPSL. The difference lies in the calculation of the relevance score between the data vector (or cluster vector) and query point. Eq. 11 shows the relevance score calculation in VPSL+, where $\Delta = s^a_{n+1}\mu_1 + s^b_{n+1}\mu_2$ and s^a_{n+1}, s^b_{n+1} are two random numbers with $s^a_{n+1} + s^b_{n+1} = s_{n+1}$. The correctness of VPSL+ is obvious because $\mathbf{p}^*_i \cdot \mathbf{T_q} - \mathbf{p}^*_j \cdot \mathbf{T_q} = \alpha\beta_{\mathbf{q}}\gamma_{\mathbf{q}}(D(\mathbf{p}_i, \mathbf{q}) - D(\mathbf{p}_j, \mathbf{q}))$ for $i, j \in [1, m], i \neq j$, i.e., the similarity between the query and data records can be compared according to the analysis in **Search** of VPSL.

$$\begin{aligned} \mathbf{p}^*_i \cdot \mathbf{T_q} &= \hat{\mathbf{p}}^a_i\widehat{\mathbf{M}}^{-1} \cdot \beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widehat{\mathbf{M}}(\bar{\mathbf{q}}^a)^\top + \hat{\mathbf{p}}^b_i\widetilde{\mathbf{M}}^{-1} \cdot \beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widetilde{\mathbf{M}}(\bar{\mathbf{q}}^b)^\top \\ &= \alpha\beta_{\mathbf{q}}\gamma_{\mathbf{q}}(||\mathbf{p}_i||^2 - 2\mathbf{p}_i\mathbf{q}^\top) + \beta_{\mathbf{q}}\gamma_{\mathbf{q}}\sum_{j=1}^{n}(r_j q_j + s_j t_j) \\ &\quad - \sum_{\mathbf{S}[j]=0} t_j r_j + \beta_{\mathbf{q}}\gamma_{\mathbf{q}}(r_{n+1} + \Delta). \end{aligned} \tag{11}$$

VI. Security Analysis

In this section, we analyze the security of VPSL and VPSL+ according to the threat model and privacy requirements seen in Section IV-B, IV-C. The data and query confidentiality as well as trapdoor unlinkability of VPSL or VPSL+ can be proved in the following theorems.

Theorem 1. VPSL is secure against the level-2 attack if $\widehat{\mathbf{M}}$ is a random invertible matrix and keeps in private.

Proof: It is obvious that our encryption is more secure than the simple encryption in ASPE [17], which has been proved to resist the level-2 attack when $\widehat{\mathbf{M}}$ is a random invertible matrix and keeps in private. Thus, VPSL is also secure against the level-2 attack. ∎

Theorem 2. VPSL is not secure against the level-3 attack if there are $(2n+2)$ data vectors $\mathcal{P}' = \{\mathbf{p}_1, \cdots, \mathbf{p}_{2n+2}\}$ linearly independent.

Proof: Assume that VPSL is attacked by a level-3 attacker who possesses the knowledge $\langle \Gamma^*, \mathcal{P}', I \rangle$, where $\mathcal{P}' = \{\mathbf{p}_1, \cdots, \mathbf{p}_{2n+2}\}$ and $I = \{\mathbf{p}^*_1, \cdots, \mathbf{p}^*_{2n+2}\}$. If the attacker can recover all data vectors P from authenticated index tree Γ^*, then VPSL is not secure against the level-3 attack. According to the index encryption, we have $\hat{\mathbf{p}}_i\widehat{\mathbf{M}}^{-1} = \mathbf{p}^*_i$ for $i \in [1, 2n+2]$, where $\hat{\mathbf{p}}_i = (r_1 - 2\alpha p_{i,1}, s_1, \cdots, r_n - 2\alpha p_{i,n}, s_n, r_{n+1} + \alpha\sum_{j=1}^{n} p^2_{i,j}, s_{n+1})$ can be expressed as $\mathbf{u}_0 - 2\alpha\mathbf{u}_i$ and $\mathbf{u}_0 = (r_1, s_1, \cdots, r_{n+1}, s_{n+1}), \mathbf{u}_i = (p_{i,1}, 0, \cdots, p_{i,n}, 0, -\frac{1}{2}||\mathbf{p}_i||^2, 0)$. Thus, $\hat{\mathbf{p}}_i\widehat{\mathbf{M}}^{-1} = \mathbf{p}^*_i$ can be rewritten as $\mathbf{u}_0\widehat{\mathbf{M}}^{-1} - 2\alpha\mathbf{u}_i\widehat{\mathbf{M}}^{-1} = \mathbf{p}^*_i$. Let $\bar{\mathbf{p}}_i$ be the $(n+1)$-dimensional vector $(p_{i,1}, \cdots, p_{i,n}, -\frac{1}{2}||\mathbf{p}_i||^2)$ and $\widehat{\mathbf{M}}_o$ be the $(n+1) \times (2n+2)$-dimensional matrix including the odd rows of $\widehat{\mathbf{M}}$. We have $\mathbf{u}_0\widehat{\mathbf{M}}^{-1} - 2\alpha\mathbf{u}_i\widehat{\mathbf{M}}^{-1} = \mathbf{u}_0\widehat{\mathbf{M}}^{-1} - 2\alpha\bar{\mathbf{p}}_i\widehat{\mathbf{M}}_o$ and Eq. 12 can be obtained by performing vector subtraction on both sides of two equations. Let A be a $(n+1) \times (n+1)$-dimensional matrix and B be a $(n+1) \times (2n+2)$-dimensional matrix such that $A = (\bar{\mathbf{p}}_2 - \bar{\mathbf{p}}_1, \bar{\mathbf{p}}_4 - \bar{\mathbf{p}}_3, \bar{\mathbf{p}}_{2n+2} - \bar{\mathbf{p}}_{2n+1})^\top, B = (\mathbf{p}^*_2 - \mathbf{p}^*_1, \mathbf{p}^*_4 - \mathbf{p}^*_3, \cdots, \mathbf{p}^*_{2n+2} - \mathbf{p}^*_{2n+1})^\top$. Eq. 12 can be expressed as $2\alpha A\widehat{\mathbf{M}}^{-1}_o = B$. Note that A is invertible since $\bar{\mathbf{p}}_2 - \bar{\mathbf{p}}_1, \cdots, \bar{\mathbf{p}}_{2n+2} - \bar{\mathbf{p}}_{2n+1}$ are linearly independent. Also, A is exposed to the attacker knowing $\mathcal{P}'$. Thus, we can compute $2\alpha\widehat{\mathbf{M}}^{-1}_o = A^{-1}B$, then obtain $2\alpha\mathbf{u}_i\widehat{\mathbf{M}}^{-1} = 2\alpha\bar{\mathbf{p}}_i\widehat{\mathbf{M}}^{-1}_o = \bar{\mathbf{p}}_i A^{-1}B$ and $\mathbf{u}_0\widehat{\mathbf{M}}^{-1} = \mathbf{p}^*_i + \bar{\mathbf{p}}_i A^{-1}B$ when $i \in [1, 2n+2]$. For each unknown data vector $\mathbf{p}_j \notin \mathcal{P}'$, we have $\mathbf{u}_0\widehat{\mathbf{M}}^{-1} - 2\alpha\mathbf{u}_j\widehat{\mathbf{M}}^{-1} = \mathbf{p}^*_i + \bar{\mathbf{p}}_i A^{-1}B - 2\alpha\bar{\mathbf{p}}_j\widehat{\mathbf{M}}^{-1}_o = \mathbf{p}^*_i + \bar{\mathbf{p}}_i A^{-1}B - \bar{\mathbf{p}}_j A^{-1}B = \mathbf{p}^*_j$. Thus, we can obtain $\hat{\mathbf{p}}_j = (\mathbf{p}^*_i - \mathbf{p}^*_j)B^{-1}A + \bar{\mathbf{p}}_i$, as $\mathbf{p}^*_i, \mathbf{p}^*_j, B^{-1}A, \bar{\mathbf{p}}_i$ are exposed to the attacker. In this way, the level-3 attacker can recover all data vectors in P, i.e., VPSL is not secure against the level-3 attack.

$$\begin{cases} 2\alpha(\bar{\mathbf{p}}_2 - \bar{\mathbf{p}}_1)\widehat{\mathbf{M}}^{-1}_o = \mathbf{p}^*_2 - \mathbf{p}^*_1; \\ 2\alpha(\bar{\mathbf{p}}_4 - \bar{\mathbf{p}}_3)\widehat{\mathbf{M}}^{-1}_o = \mathbf{p}^*_4 - \mathbf{p}^*_3; \\ \vdots \\ 2\alpha(\bar{\mathbf{p}}_{2n+2} - \bar{\mathbf{p}}_{2n+1})\widehat{\mathbf{M}}^{-1}_o = \mathbf{p}^*_{2n+2} - \mathbf{p}^*_{2n+1}; \end{cases} \tag{12}$$

∎

Theorem 3. VPSL+ is secure against the level-3 attack if the secret key $\mathcal{K} = \{\alpha, \mathbf{r}, \mathbf{s}, \mathbf{S}, \widehat{\mathbf{M}}^{-1}, \widetilde{\mathbf{M}}^{-1}\}$ keeps in private.

Proof: Assume that the knowledge of a level-3 attacker is $\langle \Gamma^*, \mathcal{P}', I \rangle$. For each data vector $\mathbf{p}_i \in \mathcal{P}'$, $\mathbf{r}, \mathbf{s}, \mathbf{S}$ are the random vectors and the attacker does not know them, he/she has to take $\mathbf{p}^a_i, \mathbf{p}^b_i$ as two random $(2n+2)$-dimensional vectors. Thus, the attacker cannot solve the plaintext data vectors based on the proof analysis in **Theorem 2**. Besides, for the linear equations constructed from all encrypted data vectors $\{\mathbf{p}^*_i = (\mathbf{p}^a_i\widehat{\mathbf{M}}^{-1}, \mathbf{p}^b_i\widetilde{\mathbf{M}}^{-1})\}_{i \in [1,m]}$, there are most $2(2n+2)^2$ unknowns in $\widehat{\mathbf{M}}, \widetilde{\mathbf{M}}$ and $2(2n+2)m$ unknowns in $\mathbf{p}^a_i, \mathbf{p}^b_i (i \in [1, m])$. Although $2(2n+2)m$ equations are given, the attacker cannot solve the matrices $\widehat{\mathbf{M}}, \widetilde{\mathbf{M}}$ as there are totally $2(2n+2)^2 + 2(2n+2)m$ unknowns. Thus, VPSL+

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

can resist the level-3 attack to guarantee index confidentiality. ■

Theorem 4. VPSL or VPSL+ ensures that plaintext query point cannot be leaked to DO or CS throughout encrypted data retrieval if DO cannot obtain $\mathbf{t}$ and CS cannot obtain $\beta_{\mathbf{q}}, \gamma_{\mathbf{q}}, \widehat{\mathbf{M}}$ or $\beta_{\mathbf{q}}, \gamma_{\mathbf{q}}, \widehat{\mathbf{M}}, \widetilde{\mathbf{M}}$.

Proof: In the process of executing TGP, DO in VPSL or VPSL+ obtains $\hat{\mathbf{q}} = \mathbf{q} + \mathbf{t} = (q_1 + t_1, \cdots, q_n + t_n)$. Since vector $\mathbf{t}$ is generated randomly and different for different queries, DO in VPSL or VPSL+ cannot know the plaintext query. Besides, due to the communications between DO and QUs on a secure channel, CS in VPSL or VPSL+ can only legally learn the trapdoor $\mathbf{T_q} = \beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widehat{\mathbf{M}}(\bar{\mathbf{q}})^\top$ or $\mathbf{T_q} = (\beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widehat{\mathbf{M}}(\bar{\mathbf{q}}^a)^\top, \beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widetilde{\mathbf{M}}(\bar{\mathbf{q}}^b)^\top)$. For VPSL, there are $(2n+2)^2$ unknowns in $\widehat{\mathbf{M}}$ and $(2n+1)$ unknowns in $\bar{\mathbf{q}}$ and $\beta_{\mathbf{q}}, \gamma_{\mathbf{q}}$ are unknowns. Although CS is given $(2n+2)$ equations, he/she cannot deduce any sensitive information as there are $(2n+2)^2 + 2n + 3$ unknowns in total. For VPSL+, there are $3(n+1)$ unknowns in $\bar{\mathbf{q}}^a, \bar{\mathbf{q}}^b$ and $2(2n+2)^2$ unknowns in $\widehat{\mathbf{M}}, \widetilde{\mathbf{M}}$, thus CS cannot deduce plaintext query point or recover $\widehat{\mathbf{M}}, \widetilde{\mathbf{M}}$, as there are only $2(2n+2)$ equations, namely the query confidentiality can be ensured. ■

Theorem 5. VPSL or VPSL+ achieves the trapdoor unlinkability for the same query point.

Proof: For the trapdoor $\mathbf{T_q} = \beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widehat{\mathbf{M}}(\bar{\mathbf{q}})^\top$ in VPSL, the randomness is introduced by the random numbers $\mu \in \mathbb{R}, \beta_{\mathbf{q}}, \gamma_{\mathbf{q}} \in \mathbb{R}^+$ and random vector $\mathbf{t} = (t_1, \cdots, t_n)$. Then, the probability of two trapdoors $\mathbf{T_{q1}}, \mathbf{T_{q2}}$ from the same query point $\mathbf{q}$ is $\mathsf{Pr}^{\mathsf{VPSL}}[\mathbf{T_{q1}} = \mathbf{T_{q2}}] = 1/|\mathbb{R}|^{n+3}$, which is close to zero. For the trapdoor $\mathbf{T_q} = (\beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widehat{\mathbf{M}}(\bar{\mathbf{q}}^a)^\top, \beta_{\mathbf{q}}\gamma_{\mathbf{q}}\widetilde{\mathbf{M}}(\bar{\mathbf{q}}^b)^\top)$ in VPSL+, the randomness originates not only from the choice of random numbers $\mu_1, \mu_2, \varepsilon_1, \varepsilon_2 \in \mathbb{R}, \beta_{\mathbf{q}}, \gamma_{\mathbf{q}} \in \mathbb{R}^+$ and random vector $\mathbf{t}$, but also from the random split in query vector encryption. $\hat{\mathbf{q}}[j]$ is split into two random values $(\hat{\mathbf{q}}^a[j], \hat{\mathbf{q}}^b[j])$ for each dimension j satisfying $\mathbf{S}[j] = 0$. Assume that $\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2$ are the same query vectors and the probability $\mathsf{Pr}[(\hat{\mathbf{q}}_1^a[j], \hat{\mathbf{q}}_1^b[j]) = (\hat{\mathbf{q}}_2^a[j], \hat{\mathbf{q}}_2^b[j])]$ is γ for each dimension j satisfying $\mathbf{S}[j] = 0$, then the probability of obtaining the same trapdoors is $\mathsf{Pr}[\mathbf{T_{q1}} = \mathbf{T_{q2}}] = 1/|\mathbb{R}|^{n+6} * \gamma^\epsilon$, where ϵ is the number of 0s in the splitting vector $\mathbf{S}$. As $\bar{\mathbf{q}}_1^a[j], \bar{\mathbf{q}}_2^a[j]$ are two random elements, the value γ is far less than $\frac{1}{2}$, thus $\mathsf{Pr}^{\mathsf{VPSL+}}[\mathbf{T_{q1}} = \mathbf{T_{q2}}] < \mathsf{Pr}^{\mathsf{VPSL}}[\mathbf{T_{q1}} = \mathbf{T_{q2}}]$ goes to zero faster. Therefore, the probability that CS obtains the same trapdoors for a query point in VPSL or VPSL+ is negligible. ■

Additionally, the correctness of search results is guaranteed by the unforgeability of MHT and the digital signature. As the root digest of MHT is signed by the secure digital signature technique such as RSA, it cannot be tampered with. Thus, the threat mainly comes from the existence of hash collision somewhere in the authenticated index tree, such that the root digest remains the same and a leaf digest is different. Since the hash function $H(\cdot)$ used in MHT is a one-way collision-resistance hash function, the probability that malicious CS forges a ciphertext or encrypted data vector such that its digest is equal to the corresponding leaf digest is negligible. Also, the change of any tree node digest stored in CS will change its root digest. Thus, it is impossible for the situation that the root digest remains the same and a leaf digest is different. That is to say, the search results are correct if $H(\cdot)$ is a one-way collision-resistance hash function.

VII. Performance Analysis

In this section, we analyze the performance of our proposed schemes theoretically and experimentally by comparing them with the privacy-preserving secure kNN computation scheme (SkNN [18]) which is the basis of our proposed schemes. For the theoretical analysis, we mainly concentrate on communication and computation overheads. After that, we perform experiments with real-world datasets to test search accuracy and efficiency of VPSL or VPSL+. The notations used in the theoretical analysis are described in TABLE II.

TABLE II: Notation definitions for performance analysis

Notations	Descriptions
m	size of dataset F
m'	number of non-leaf node in Γ^*
n	number of attributes in dataset F
l	$l = m + n + 1$
N	width of authenticated index tree Γ^*
d	length of expended data vector, $d = 2n + 2$
c	average storage cost of an encrypted data record
λ	number of bits occupied by a floating number
η	number of nodes encountered during search, $\eta \ll m$
$\mathbb{D}$	storage cost of the digests for all nodes in Γ^*
$\mathrm{T_{PRF}}$	time taken to compute a pseudo-random function
$\mathrm{T_{SPL}}$	time taken to randomly split a n-dimensional vector
$\mathrm{T^*_{ADD}}$	time taken to add two d-dimensional vectors
$\mathrm{T'_{ADD}}$	time taken to add two n-dimensional vectors
$\mathrm{T^*_{DOT}}$	time spending on dot product of two d-dimensional vectors
$\mathrm{T'_{DOT}}$	time spending on dot product of two n-dimensional vectors
$\mathbb{G}_1, \mathbb{G}_2$	storage cost of VO in VPSL and VPSL+, respectively
$\mathbb{P}$	time taken to authenticate index tree, $\mathbb{P} = (m + m')\mathrm{T_{PRF}}$

A. Theoretical Analysis

The communication and computation overheads of SkNN, VPSL and VPSL+ are shown in TABLE III.

Communication overhead. In SkNN, the communication overheads of CS, DO, QU are mainly introduced by returning top-k search results, uploading encrypted data records and sending $\mathbf{x}, \mathbf{Y}$ to QU, sending the trapdoor to CS and $\hat{\mathbf{q}}$ to DO, respectively. In VPSL, DO also needs to upload encrypted cluster vectors (i.e., $m'd\lambda$) and the digests of all nodes in Γ^* (i.e., $\mathbb{D} = 128(m+m')$ bit if we use SHA1 to generate the digests); CS also needs to return auxiliary information VO (i.e., $\mathbb{G}_1$). In addition, due to the random splitting technique, each encrypted data vector (or trapdoor) in VPSL+ consists of two d-dimensional vectors. Thus, the communication overheads of DO, QU and CS in VPSL+ are $2(l + m')d\lambda + \mathbb{D}$, $(2d + n)\lambda$ and $kc + \mathbb{G}_2$ respectively, where $\mathbb{G}_2 = \mathbb{G}_1 + (k + \mathrm{L}_p)d$ and L_p denotes the length of the search path. From the TABLE III, we find that some communication resources need to be sacrificed for achieving efficient ciphertext search, result verification and greater security.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

TABLE III: Theoretical communication and computation: A comparative summary

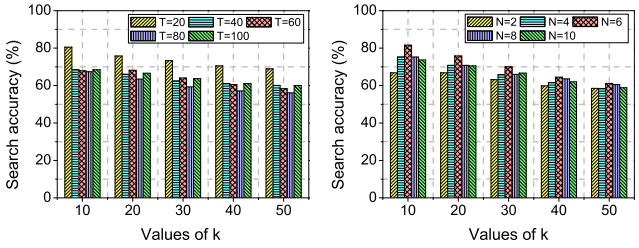
Schemes	Communication			Computation			
	CS	DO	QU	**BulidTree**	**TrapGen**	**Search**	**Verify**
SkNN	kc	$ld\lambda$	$(d+n)\lambda$	$md\mathrm{T}^{*}_{\mathrm{DOT}}$	$2(\mathrm{T}^{*}_{\mathrm{ADD}}+\mathrm{T}'_{\mathrm{ADD}})+2d\mathrm{T}'_{\mathrm{DOT}}$	$m\mathrm{T}^{*}_{\mathrm{DOT}}$	N/A
VPSL	$kc+\mathbb{G}_1$	$(l+m')d\lambda+\mathbb{D}$	$(d+n)\lambda$	$(m+m')d\mathrm{T}^{*}_{\mathrm{DOT}}+\mathbb{P}$	$2(\mathrm{T}^{*}_{\mathrm{ADD}}+\mathrm{T}'_{\mathrm{ADD}})+2d\mathrm{T}'_{\mathrm{DOT}}$	$\eta\mathrm{T}^{*}_{\mathrm{DOT}}$	$(2\mathrm{L}_p+k)\mathrm{T}_{\mathrm{PRF}}$
VPSL+	$kc+\mathbb{G}_2$	$2(l+m')d\lambda+\mathbb{D}$	$(2d+n)\lambda$	$2(m+m')d\mathrm{T}^{*}_{\mathrm{DOT}}+\mathbb{P}$	$4\mathrm{T}^{*}_{\mathrm{ADD}}+\mathrm{T}'_{\mathrm{ADD}}+4d\mathrm{T}'_{\mathrm{DOT}}+\mathrm{T}_{\mathrm{SPL}}$	$2\eta\mathrm{T}^{*}_{\mathrm{DOT}}$	$(2\mathrm{L}_p+k)\mathrm{T}_{\mathrm{PRF}}$

Computation overhead. For expression simplicity, we set $\mathrm{T}^{*}_{\mathrm{ADD}}$ as the time spending on dot product of two d-dimensional vectors, thus the multiplication time of a d-dimensional vector and a $d \times d$-dimensional matrix can be viewed as d $\mathrm{T}^{*}_{\mathrm{ADD}}$. Taking no consideration for the time spending on k-means clustering and signature generation, the computation overhead of VPSL in **BulidTree** is composed of encrypting the index tree (i.e., $(m+m')d\mathrm{T}^{*}_{\mathrm{DOT}}$) and authenticating the index tree (i.e., $\mathbb{P}=(m+m')\mathrm{T}_{\mathrm{PRF}}$). As VPSL+ splits each data vector (or cluster vector) into two vectors, the computation overhead of VPSL+ is $2(m+m')d\mathrm{T}^{*}_{\mathrm{DOT}}+\mathbb{P}$ by ignoring the vector split time, which is more than that of VPSL. In addition, both of them are greater than that of SkNN which only needs to encrypt data vectors. In **TrapGen**, SkNN and VPSL nearly have the same computation overhead because of executing the similar trapdoor generation protocol. For higher security, VPSL+ splits the query point (i.e., $\mathrm{T}_{\mathrm{SPL}}$), performs vector addition operation (i.e., $\mathrm{T}'_{\mathrm{ADD}}$), computes $\dot{\mathbf{x}}, \dot{\mathbf{Y}}, \ddot{\mathbf{x}}, \ddot{\mathbf{Y}}$ (i.e, $2\mathrm{T}^{*}_{\mathrm{ADD}}+2d\mathrm{T}'_{\mathrm{DOT}}$) and generates the trapdoor $\mathbf{T_q}=(\mathbf{T}^{a}_{\mathbf{q}},\mathbf{T}^{b}_{\mathbf{q}})$ (i.e., $2\mathrm{T}^{*}_{\mathrm{ADD}}+2d\mathrm{T}'_{\mathrm{DOT}}$). Thus, the computation overhead of VPSL+ is almost twice that of SkNN and VPSL. Despite much time taken in **BulidTree**, the computation overheads of VPSL and VPSL+ in **Search** are much less than that of SkNN due to $\eta \ll m$. Besides, in **Verify**, VPSL or VPSL+ takes $(2\mathrm{L}_p+k)\mathrm{T}_{\mathrm{PRF}}$ to finish result verification, which is mainly related to the number of search results and the length of search path.

B. Experimental Tests

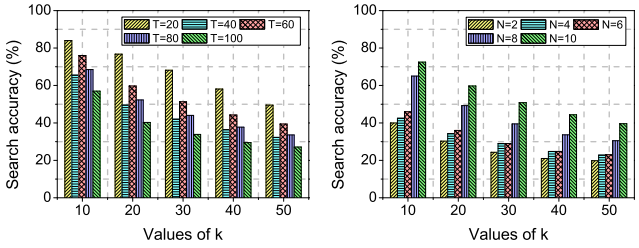
We conduct experiments on a 64-bit Window Server with two 3.60GHz Intel(R) Core(TM) i7-9700K CPU by using Python programming language with sklearn library. The test datasets of search accuracy stem from the Breast Cancer Wisconsin (Diagnostic) Data Set[4] (called BCW) and the Human Activity Recognition Using Smartphones Data Set[4] (called HARS). BCW contains 569 data records with 31 attributes (ignoring the ID number attribute). HARS contains 10409 data records with 561 attributes. The test datasets of efficiency stem from HARS and Epileptic Seizure Recognition Data Set[4] (called ESR) which contains 11500 data records with 178 attributes. In addition, we use SHA1 to generate each node's digest.

Search Accuracy. To evaluate the search accuracy of top-k search results, we use the most common measure $P_k = k'/k$ [11], where k' is the number of real top-k query results returned by CS. According to the relevance comparison of VPSL or VPSL+, the search accuracy is not affected by the added random numbers or encryption, but affected by k-means clustering. The reason is that search results come from the best matching cluster, but the data records in real top-k may not in this cluster. To demonstrate this, we randomly select 550 and 10000 instances out BCW and HARS respectively to complete their k-means clustering, then use 19 instances to test their average search accuracy.



(a) $N=2$ (b) $T=50$

Fig. 7: Search accuracy of VPSL or VPSL+ on BCW.



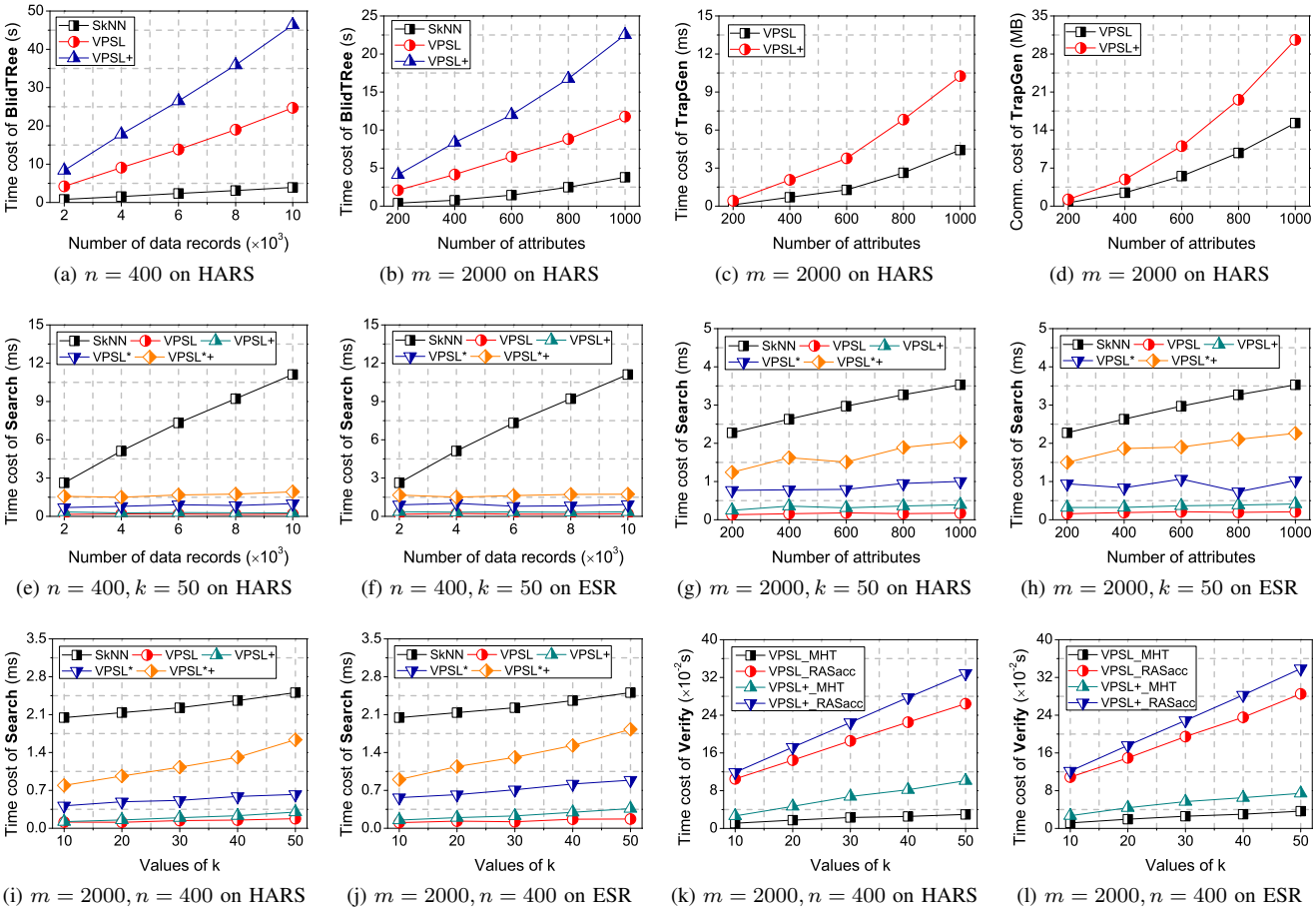
(a) $N=10$ (b) $T=50$

Fig. 8: Search accuracy of VPSL or VPSL+ on HARS.

By varying the values of $k \in [10,50]$, threshold $T \in [20,100]$ and width $N \in [2,10]$, we plot the search accuracy of VPSL (or VPSL+) on BCW and HARS in Figs. 7, 8, respectively. We find that the threshold T and width N affect the search accuracy, which is about 80% when $T=20, N=2, k=10$ for BCW and about 82% when $T=20, N=10, k=10$ for HARS. Then, we compare the search accuracy of VPSL (or VPSL+) and SkNN on BCW when $T=20, N=6$ and on HARS when $T=20, N=10$ shown in TABLE IV, V. SkNN not using the k-means clustering technique computes and compares the relevance scores of query and data records with the same method as VPSL; thus it can obtain search results without accuracy loss. In VPSL (or VPSL+), the larger k incurs greater accuracy loss. Therefore, we should select proper T, N, k so that the accuracy loss caused by clustering is acceptable. If there is no such T, N, k, we search each subtree of the root node instead of searching from the root node to retrieve some data records in the real top-k at the expense of some computation cost. Let VPSL*

[4]https://archive.ics.uci.edu/ml/datasets

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing



(a) $n = 400$ on HARS (b) $m = 2000$ on HARS (c) $m = 2000$ on HARS (d) $m = 2000$ on HARS

(e) $n = 400, k = 50$ on HARS (f) $n = 400, k = 50$ on ESR (g) $m = 2000, k = 50$ on HARS (h) $m = 2000, k = 50$ on ESR

(i) $m = 2000, n = 400$ on HARS (j) $m = 2000, n = 400$ on ESR (k) $m = 2000, n = 400$ on HARS (l) $m = 2000, n = 400$ on ESR

Fig. 9: Practical performance analysis of our proposed schemes.

(or VPSL*+) be VPSL (or VPSL+) with this retrieval way. The search accuracy of VPSL* (or VPSL*+) in TABLE IV, V shows that this retrieval way is feasible to improve accuracy.

TABLE IV: Search accuracy on BCW when $N = 6, T = 20$

k	10	20	30	40	50
SkNN	100	100	100	100	100
VPSL/VPSL+	84.74	79.21	73.33	67.89	62.26
VPSL*/VPSL*+	97.37	96.84	95.96	95.92	95.37

TABLE V: Search accuracy on HARS when $N = 10, T = 20$

k	10	20	30	40	50
SkNN	100	100	100	100	100
VPSL/VPSL+	84.00	76.75	68.17	58.13	49.5
VPSL*/VPSL*+	88.00	84.75	80.67	78.50	74.40

Efficiency. We set $N = 6, T = 100$ to test the index construction time, trapdoor generation time, search time and result verification time. We test the index construction time, trapdoor generation time on one dataset (i.e., HARS). We conduct 20 random queries to test the average search time on HARS and ESR to show the impact of different datasets on query efficiency. We test the result verification efficiency by comparing the Merkle hash verification with another verification mechanism (i.e., RSA accumulator), which has been wildly used to achieve result verification on the privacy-preserving keyword search. The implementation details of RSA accumulator are shown as follows:

RSA accumulator. We randomly generate two 128-bit primes p, q so that $p = 2p' + 1, q = 2q' + 1$ and p', q' are also two primes. Besides, the generator g of the cyclic group G is 2. To construct a RSA accumulator for index set $\mathcal{I} = \{I_1, \cdots, I_m\}$, we first compute $H(i, I_i) = H(i|x_{i,1}| \cdots |x_{i,2n+2})$ to convert each index $I_i = (x_{i,1}, \cdots, x_{i,2n+2})$ into $\{0,1\}^*$, where $H(\cdot)$ is SHA1 and $''|''$ represents concatenation operation, then generate an accumulator $\mathbf{A}_{\mathcal{I}} = \prod_{i=1}^{m} g^{\mathcal{P}(H(i,I_i))} \bmod \mathrm{N}$, where $\mathrm{N} = pq$ and $\mathcal{P}(H(i, I_i))$ is a randomly chosen prime generated by $H(i, I_i)$. For the top-k search results $\mathcal{R} = \{I_{1'}, \cdots, I_{k'}\}$, CS generates a proof $\mathrm{P}_f = g^{\prod_{i \notin topk} \mathcal{P}(H(i,I_i))} \bmod \mathrm{N}$, and sends it together with the search results $\mathcal{R}$ to DU. Here, $topk = \{1', \cdots, k'\}$. Subsequently, the verification is carried out by checking $\mathbf{A}_{\mathcal{I}} \stackrel{?}{=} (\mathrm{P}_f)^{\prod_{i'=1}^{k} \mathcal{P}(H(i',I_{i'}))} \bmod \mathrm{N}$.

Index construction time. The authenticated index tree construction of VPSL or VPSL+ consists of k-means clustering, index encryption and authentication, which is influenced by the threshold T, width N, number of data records m and number of attributes n. TABLE VI shows that the **BuildTree** time of VPSL and VPSL+ reduces slightly with the growth of T.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3031209, IEEE Transactions on Cloud Computing

Figs. 9a, 9b show that the index construction time of SkNN[5], VPSL and VPSL+ grows with m and n. Besides, the index construction overhead of VPSL+ is twice that of VPSL, and both of them are much higher than that of SkNN. Fortunately, the index construction is a one-time operation for static data.

TABLE VI: Time cost of **BuildTree** by varying threshold and width on HARS (seconds)

VPSL				VPSL+			
T	Time (s)	N	Time (s)	T	Time (s)	N	Time (s)
100	4.17	6	4.15	100	8.31	6	7.99
300	3.79	12	4.22	300	7.58	12	8.57
500	3.68	18	4.41	500	7.50	18	9.48
700	3.34	24	4.47	700	6.99	24	9.84
900	3.28	30	4.46	900	6.99	30	9.29

Trapdoor generation time. Since SkNN and VPSL have the same computation and communication costs in **TrapGen**, which are both related to the number of attributes n seen in TABLE III, we just analyze the computation and communication costs of trapdoor generation for VPSL and VPSL+ by varying n. Figs. 9c, 9d show that the computation and communication costs of VPSL and VPSL+ super-linearly increase with the number of attributes and VPSL+ takes almost twice computation and communication costs to generate a trapdoor by comparing with VPSL, which is consistent with the theoretical analysis. In addition, the communication cost is acceptable for fewer attributes. When the number of attributes $n = 1000$, the communication costs of VPSL and VPSL+ are about 15 MB and 30 MB, respectively.

Search time. Figs. 9e, 9f, 9g, 9h, 9i, 9j show that no matter in HARS or ESR, the search time of SkNN is much more than that of VPSL, VPSL+, VPSL* and VPSL*+, and more data records will lead to a greater difference in their search time. VPSL always takes the shortest search time and the search time of VPSL+ is slightly longer than that of VPSL. But anyway, VPSL+ is still very efficient and more suitable for real-time applications. The search time of VPSL* (or VPSL*+) is about six times that of VPSL (or VPSL+), but compared with SkNN, it is acceptable especially for massive datasets. In addition, Figs. 9e, 9f demonstrate that the tree structure makes the search time of VPSL, VPSL+, VPSL* and VPSL*+ not sensitive to the number of data records m. Figs. 9i, 9j show that the search time of VPSL*+ increases with the values of k faster than that of SkNN; thus we should choose k as small as possible.

Result verification time. Figs. 9k, 9l express the result verification time of VPSL and VPSL+ based on MHT and RSA accumulator [19] (RSAacc, for short) in different values of k when $m = 2000, n = 400$. We notice that the RSAacc (or MHT) result verification time of our schemes almost linearly grows with k. The Merkle verification time is much less than the RSAacc verification time. That is why we choose MHT structure rather than RSA accumulator to achieve verifiability. Besides, VPSL+ takes more computation cost to verification than VPSL, since for each node on the search path, VSPL+ needs to hash two $(2n + 2)$-dimensional sub-indexes, while VPSL only needs to hash one $(2n + 2)$-dimensional index.

[5]The index construction of SkNN consists of index encryption.

VIII. Conclusion

In this paper, we first proposed VPSL for cloud-assisted Internet of Things. In VPSL, we constructed an authenticated index tree based on the k-means clustering technique, Merkle hash tree and ASPE to achieve efficient ciphertext retrieval and efficient result verification, respectively. In addition, a trapdoor generation protocol was designed to obtain the trapdoor with limited key disclosure and without revealing plaintext query points to CS or DO. Then, VPSL+ was proposed by introducing the random splitting technique to resist the level-3 attack. Performance evaluation using real-world datasets demonstrated the actual performance of VPSL or VPSL+ in terms of search accuracy and efficiency. In future work, we aim to offer a dynamic update function.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61702404, No. 61702105), the Fundamental Research Funds for the Central Universities (No. JB191506), the National Natural Science Foundation of Shaanxi Province (No. 2019JQ-005).

References

[1] M. Wen, R. Lu, J. Lei, H. Li, X. Liang, and X. Shen, "Sesa: An efficient searchable encryption scheme for auction in emerging smart grid marketing," *Security and Communication Networks*, vol. 7, no. 1, pp. 234–244, 2014.

[2] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Secure data sharing and searching at the edge of cloud-assisted internet of things," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 34–42, 2017.

[3] S. Xu, Y. Li, R. Deng, Y. Zhang, X. Luo, and X. Liu, "Lightweight and expressive fine-grained access control for healthcare internet-of-things," *IEEE Transactions on Cloud Computing*, 2019.

[4] W. A. Amiri, M. Baza, K. Banawan, M. Mahmoud, W. Alasmary, and K. Akkaya, "Privacy-preserving smart parking system using blockchain and private information retrieval," *arXiv preprint arXiv:1904.09703*, 2019.

[5] S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2101–2113, 2018.

[6] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symposium on Security and Privacy (SP'00)*. IEEE, 2000, pp. 44–55.

[7] H. Cheng, X. Zhang, J. Yu, and F. Li, "Markov process-based retrieval for encrypted jpeg images," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 1, 2016.

[8] R. Poddar, T. Boelter, and R. A. Popa, "Arx: A strongly encrypted database system." *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 591, 2016.

[9] Y. Guo, X. Yuan, X. Wang, C. Wang, B. Li, X. Jia *et al.*, "Enabling encrypted rich queries in distributed key-value stores," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1283–1297, 2018.

[10] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. International Conference on Applied Cryptography and Network Security (ACNS'04)*. Springer, 2004, pp. 31–45.

[11] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222–233, 2013.

[12] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 312–325, 2015.

[13] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. ACM Symposium on Information, Computer and Communications Security (AsiaCCS'13)*. ACM, 2013, pp. 71–82.

[14] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE transactions on parallel and distributed systems*, vol. 27, no. 2, pp. 340–352, 2015.

[15] C. Chen, X. Zhu, P. Shen, J. Hu, S. Guo, Z. Tari, and A. Y. Zomaya, "An efficient privacy-preserving ranked keyword search method," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 951–963, 2015.

[16] J. Yuan, S. Yu, and L. Guo, "Seisa: Secure and efficient encrypted image search with access control," in *Proc. IEEE Conference on Computer Communications (INFOCOM'15)*. IEEE, 2015, pp. 2083–2091.

[17] W. K. Wong, D. W. l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. ACM International Conference on Management of Data (SIGMOD'09)*. ACM, 2009, pp. 139–152.

[18] Y. Zhu, R. Xu, and T. Takagi, "Secure k-nn computation on encrypted cloud data without sharing key with query users," in *Proc. International Workshop on Security in Cloud Computing (SCC'13)*. ACM, 2013, pp. 55–60.

[19] Q. Liu, X. Nie, X. Liu, T. Peng, and J. Wu, "Verifiable ranked search over dynamic encrypted data in cloud computing," in *Proc. IEEE/ACM International Symposium on Quality of Service (IWQoS'17)*. IEEE, 2017, pp. 1–6.

[20] W. Sun, X. Liu, W. Lou, Y. Hou, and H. Li, "Catch you if you lie to me: efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," 05 2015.

[21] Z. Wan and R. H. Deng, "Vpsearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1083–1095, 2016.

[22] Y. Miao, J. Ma, X. Liu, J. Zhang, and Z. Liu, "Vkse-mo: verifiable keyword search over encrypted data in multi-owner settings," *Science China Information Sciences*, vol. 60, no. 12, pp. 122 105:1–122 105:15, 2017.

[23] M. Cao, L. Wang, Z. Qin, and C. Lou, "A lightweight fine-grained search scheme over encrypted data in cloud-assisted wireless body area networks," *Wireless Communications and Mobile Computing*, vol. 2019, pp. 9 340 808:1–9 340 808:12.

[24] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *Proc. IEEE Conference on Computer Communications (INFOCOM'15)*. IEEE, 2015, pp. 2110–2118.

[25] G. Sen Poh, J. J. Chin, W. C. Yau, K. K. R. Choo, and M. S. Mohamad, "Searchable symmetric encryption: Designs and challenges," *Acm Computing Surveys*, vol. 50, no. 3, pp. 40.1–40.37.

[26] R. Zhang, R. Xue, and L. Liu, "Searchable encryption for healthcare clouds: A survey," *IEEE Transactions on Services Computing*, pp. 1–1.

[27] Y. Wang, J. Wang, and X. Chen, "Secure searchable encryption: a survey," *Journal of Communications & Information Networks*, vol. 1, no. 4, pp. 52–65.

[28] J. Li and L. Zhang, "Attribute-based keyword search and data access control in cloud," in *Proc. International Conference on Computational Intelligence and Security (CIS'14)*. IEEE, 2014, pp. 382–386.

[29] R. E. Overill, "Review: foundations of cryptography, volume ii: basic applications," *Journal of Logic & Computation*, no. 3, p. 3, 2005.

[30] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annual International Cryptology Conference (CRYPTO'91)*. Springer, 1991, pp. 129–140.

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

[32] R. Bost, P.-A. Fouque, and D. Pointcheval, "Verifiable dynamic symmetric searchable encryption: Optimality and forward security." *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 62, 2016.

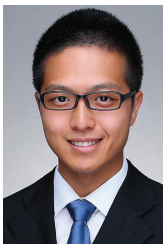
[33] J. Zhu, Q. Li, C. Wang, X. Yuan, Q. Wang, and K. Ren, "Enabling generic, verifiable, and secure data search in cloud services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 8, pp. 1721–1735, 2018.

[34] Y. Guo, C. Zhang, and X. Jia, "Verifiable and forward-secure encrypted search using blockchain techniques," in *Proc. IEEE International Conference on Communications (ICC'20)*. IEEE, 2020, pp. 1–7.

[35] H. Cheng, H. Wang, X. Liu, Y. Fang, M. Wang, and X. Zhang, "Person re-identification over encrypted outsourced surveillance videos," *IEEE Transactions on Dependable and Secure Computing*, 2019.

Qiuyun Tong received the B.E. degree with the Department of Information and Computing Science from Shaanxi Normal University, Xi'an, China, in 2019. Now she is pursuing her M.E degree with the Department of Cyber Engineering from Xidian University, Xi'an, China. Her research interests include information security and applied cryptography.

Yinbin Miao (M'18) received the B.E. degree with the Department of Telecommunication Engineering from Jilin University, Changchun, China, in 2011, and Ph.D. degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China, in 2016. He is also a postdoctor in Nanyang Technological University from September 2018 to September 2019. He is currently a Lecturer with the Department of Cyber Engineering in Xidian University, Xi'an, China. His research interests include information security and applied cryptography.



Ximeng Liu (M'16) received the B.E. degree with the Department of Electronic Engineering from Xidian University, Xi'an, China, in 2010 and Ph.D. degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China in 2015. He is currently a post-doctoral with the Department of Information System, Singapore Management University, Singapore. His research interests include applied cryptography and big data security. He is a member of the IEEE.

Kim-Kwang Raymond Choo (SM'15) currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), the UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty in 2018, the British Computer Society's 2019 Wilkes Award Runner-up, the 2019 EURASIP JWCN, IEEE TrustCom 2018 and ESORICS 2015 Best Paper awards, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He is also an Australian Computer Society Fellow.

Robert H. Deng (F'16) is AXA Chair Professor of Cybersecurity and Professor of Information Systems in the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He has served on the editorial boards of many international journals, including TFIS, TDSC. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017). He is a fellow of the IEEE.

Hongwei Li (M'12) received the Ph.D. degree in computer software and theory from the University of Electronic Science and Technology of China, Chengdu, China, in 2008. He is currently a professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include network security, applied cryptography, and trusted computing. He is a member of IEEE, a member of China Computer Federation and a member of China