Peer-to-Peer Networking and Applications (2020) 13:412–423
https://doi.org/10.1007/s12083-019-00810-7

Achieve privacy-preserving simplicial depth query over collaborative cloud servers

Hassan Mahdikhani[1] · Rasoul Shahsavarifar[1] · Rongxing Lu[1] · David Bremner[1]

Received: 8 May 2019 / Accepted: 21 August 2019 / Published online: 30 August 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

The *simplicial depth* (SD) of a query point $q \in \mathbb{R}^d$ with respect to a dataset $S \subset \mathbb{R}^d$ is defined based on counting all $(d+1)$-dimensional *simplices* obtained from S that contain q. The simplicial depth is a ranking function which is frequently used in order to sort a multivariate dataset. In the higher dimension d, no better algorithm is known than the brute force method which takes $\Theta(n^{d+1})$ time, where $|S| = n$. Unfortunately, in contrast to the many advantages that have been previously identified by research studies, this depth function requires a massive amount of computation particularly for higher dimensional datasets. This challenge could be overcome by offloading the computation to cloud servers. However, delegating simplicial depth queries to not fully trusted cloud servers would be a source of serious security breaches and privacy issues. Therefore, in this paper, we target the privacy-preserving simplicial depth query over collaborative cloud servers. To this end, two resource-abundant cloud servers will be employed to perform such time consuming computation while maintaining the user's privacy. Security analysis shows our proposed scheme achieves privacy-preserving requirements. In addition, some experiments based on a dataset generated by normal distribution are conducted, and the results validate the efficiency and practicality of our proposed scheme. Although this work only focuses on the planar case, our proposed scheme can be extended into higher dimension cases without significant alterations.

Keywords Computational geometry · Data depth · Simplicial depth · Privacy-preserving · Homomorphic encryption · Collaborative cloud · Computation offloading

1 Introduction

The rank statistic tests play an important role in univariate non-parametric statistics. If one attempts to generalize the rank tests to the multivariate case, the problem of defining a multivariate order statistic will occur. One approach to overcome this problem is to use the notion of data depth. A depth function, as a tool in non-parametric multivariate data analysis, measures the centrality of a point in a given dataset. In other words, it indicates how deep a point is located with respect to the dataset. Applying a data depth on a dataset generates a partial ordered set (poset) of the data points. Over the last decades, various notions of data depth such as *halfspace depth* (Hotelling, 1929, [18, 38]; Tukey, 1975, [41]), *simplicial depth* (Liu, 1990, [23]) *Oja depth* (Oja, 1983, [30]), *regression depth* (Rousseeuw and Hubert, 1999, [35]) have emerged as powerful tools for non-parametric multivariate data analysis. Most of them have been defined to solve specific problems in data analysis. They are different in application, definition, and the geometry of their central regions (regions with the maximal depth). Some notable research on the algorithmic aspects of planar data depth can be found in [4, 8, 25]. Among all of these depth functions, we focus on the simplicial depth [23] because of its various applications in statistics and data analysis. The simplicial depth of a query point $q \in \mathbb{R}^d$ with respect to a given dataset $S \subset \mathbb{R}^d$

✉ Rongxing Lu
rlu1@unb.ca

Hassan Mahdikhani
hmahdikh@unb.ca

Rasoul Shahsavarifar
ra.shahsavari@unb.ca

David Bremner
bremner@unb.ca

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

is defined by counting all simplices[1] in S that contain q. This value can be normalized using the normalization factor $1/\binom{|S|}{d+1}$. Although there is an $O(n \log n)$ algorithm in $\mathbb{R}^2$, an $O(n^2)$ algorithm in $\mathbb{R}^3$, and an $O(n^4)$ algorithm in $\mathbb{R}^4$, the brute force method which takes $\Theta(n^{d+1})$ time is the best available algorithm for computing simplicial depth in $\mathbb{R}^d$. It is generally understood that any superlinear algorithm is impractical for sufficiently large dataset S. As the dimension d grows, the threshold at which local computations become impractical shrinks rapidly. It is thus worth investigating how expensive simplicial depth calculations can be offload to the large scale computational resources available in the cloud.

Cloud computing allows data owners to handle extremely complex computations on massive amounts of data in a near real-time manner with substantially reduced cost. In other words, instead of locally processing high-dimensional datasets and being impeded by technical barriers, companies have an opportunity to outsource their computations to abundant cloud computing peers [19]. Along with clear technical and economic benefits, such offloading brings privacy concerns. Unless addressed, these concerns would deter data owners from outsourcing their computations [34]. Although applying legacy data encryption would alleviate security and privacy concerns, it does not support logical comparison and mathematical calculation over ciphertexts. In order to obtain the corresponding results in the plaintext mode, cloud servers are required to enforce desired computations on given encrypted data. To do this, Homomorphic Encryption (HE) is a generally accepted method. HE diminishes privacy concerns by enabling the cloud servers to carry out computation on ciphertext, and preserves the crypto-ecosystem properties. It is worth noting that HE schemes are broadly categorized into three types: partially, somewhat, and fully [1]. Each of which has different limitations and strengths.

In this paper, aiming at addressing the above challenges, we would like to apply HE techniques to achieve privacy preserving simplicial depth computation over collaborative cloud servers. As far as we know, this paper is the first study to achieve those goals simultaneously. To sum up, the main contributions of this paper are three-fold:

- We propose a privacy preserving simplicial depth query ($PSDQ$) scheme, which securely offloads required computation to cloud servers. To this end, BFV homomorphic encryption [13] is used to evaluate the corresponding mathematical circuits of the simplicial depth queries. The optimal security parameters of BFV have been obtained to support the multiplicative depth=3 in the proposed $PSDQ$.
- Although our proposed $PSDQ$ can undertake the query processing with an acceptable execution time taking some *preprocessing steps* will improve its performance. In the improved-$PSDQ$, the underlying time consuming determinant calculations have been precomputed by the query processing server. Our evaluations show the improved version will outperform the basic $PSDQ$. It should be noted that precomputing determinants and their squared values will significantly enhance response time when datasets with large number of samples are offloaded.
- We do thorough experiments to evaluate the performance of $PSDQ$ in dealing with floating point values. In $PSDQ$, scaling factor φ to handle floating-point datasets has been introduced. Corresponding contour diagrams for specific scale factors have been included to visualize the differences. Moreover, two dissimilarity metrics along with an accuracy measures are introduced in order to investigate the correctness of results.

The remainder of the paper is structured as follows. In Section 2, related work is discussed. Section 3 contains some preliminaries for this study. The system and security models along with design goals are introduced in Section 4 followed by our proposed $PSDQ$ in Section 5. The security analysis of $PSDQ$ and the detailed experimental results are organized in Sections 6 and 7, respectively. Finally, conclusion and future work are discussed in Section 8.

2 Related work

In the extant privacy-preserving literature, various solutions have be discussed to enable computational geometry algorithms over encrypted datasets. A short list of the most important and well-known privacy preserving relevant studies could be identified as follows: multivariate statistical estimation [27, 40, 43], range counting [43], convex hull computation [15], and related geometric algorithms [12, 22].

More specifically, the following outcomes could be extracted from these aforementioned studies. Regarding multivariate statistical estimation, in [27, 43], the authors have investigated the possibility of applying fully homomorphic encryption for large scale statistical analysis. As another work in this domain, [40] has demonstrated that for a given distribution and a class of statistical estimators, there exists a differentially private estimator with the same distribution. Privacy preserving methods have been applied into the area of computational geometry. In this regard, convex hull, which is one of the most explored problems in computational geometry, is studied in [15]. The authors presented a secure multi-party computation (SMC) protocol to compute the convex hull of a given set of planar points

[1] A simplex in $\mathbb{R}$ is a line segment, in $\mathbb{R}^2$ is a triangle, in $\mathbb{R}^3$ is a tetrahedron, etc.

with k computational parties in a privacy preserving manner. The authors in [12] also utilized a SMC protocol to present a privacy preserving optimal data-oblivious algorithm to construct the planar convex hull of a geographic dataset. Additionally, for a set of n points, optimal algorithms for well-separated pair decomposition, compressed quadtree construction, closest pairs, and all nearest neighbor finding are addressed in [12]. Privacy preserving computation on conic sections, as a branch of computational geometry problems, has been addressed in [22]. In particular, the authors presented secure efficient constant-round protocols for solving point-included and line-intersected problems on conic sections in a semi-honest model. To achieve privacy protection, they demonstrated a secure two-party computation model by adopting the following building blocks: secure two-party computation, Paillier HE scheme [31], and secure scalar product protocol.

The present work is related to both computational geometry and (broadly speaking) multivariate statistical estimation, but to the best of our knowledge is the first work which focuses on the privacy preserving computation of simplicial depth. Our scheme accepts the encrypted datasets and securely performs simplicial depth query that can protect not only the datseset but also the query.

3 Preliminaries

In this section, we give an overview of simplicial depth, and recall the Homomorphic Encryption scheme [36], which will serve as the basis of our proposed scheme.

3.1 Simplicial depth calculation

The simplicial depth of a query point $q \in \mathbb{R}^d$ with respect to $S = \{x_1, ..., x_n\} \subset \mathbb{R}^d$ is defined as the total number of the closed simplices formed by data points that contain q. This definition can be given by (1).

$$SD(q; S) = \mu \sum\nolimits_{(x_1,...,x_{d+1})} I(q \in Conv[x_1, ..., x_{d+1}]), \tag{1}$$

where $\mu = 1/\binom{n}{d+1}$ is the normalization factor, the convex hull $Conv[x_1, ..., x_{d+1}]$ is a closed simplex formed by $d+1$ points of S, and I is the indicator function. For $S = \{a, b, c, d, e\}$ in $\mathbb{R}^2$, Fig. 1 illustrates that $SD(q_1; S) = 3/10$, whereas $SD(q_2; S) = 4/10$.

The simplicial depth of $q \in \mathbb{R}^d$ with respect to the distribution function F is defined as follows:

$$SD(q; F) = P_F(q \in Simplex[x_1, \ldots, x_{d+1}]),$$

where $Simplex[x_1, \ldots, x_{d+1}]$ is a random simplex of i.i.d observations $x_1, \ldots, x_{d+1}$ from F, and P_F is representing the probability corresponding to F.

Liu [24] proved that the simplicial depth is affinely invariant. Depending on the distribution of data points, the simplicial depth has completely different characteristics. For a *Lebesgue-continuous* distribution [16], simplicial depth changes continuously (Theorem 2 in [23]), decreases monotonically on the rays, and has a unique central region [23, 29]. Furthermore, the contours defined by simplicial depth are nested (Theorem 3 in [23]). However, if the distribution is discrete, these characteristics are not necessary applicable [44]. Unlike the contours of halfspace depth which are convex, the contours of simplicial depth are only starshaped (Section 2.3.3 in [29]). It is known that the breakdown point of the simplicial median (i.e. the point with maximal simplicial depth) is always worse than the breakdown point of halfspace median [9]. The behaviour of simplicial depth contours is not as nice as the behaviour of half-space depth contours [17, 28]. As an example, Fig. 2 illustrates that some of the simplicial depth contours may not be nested. It can be seen that the contour enclosing all points of depth $10/20 = 0.5$ and up is not surrounded by the contour enclosing depth of $8/20 = 0.4$ and up.

Simplicial depth is widely studied in the literature. Some results regarding the simplicial depth can be listed as follows.

- The simplicial depth of a query point in $\mathbb{R}^2$ can be computed using an optimal algorithm which takes $\Theta(n \log n)$ time [2, 3, 20].
- The simplicial depth of a point in $\mathbb{R}^3$ can be computed in $O(n^2)$, and in $\mathbb{R}^4$ the fastest known algorithm needs

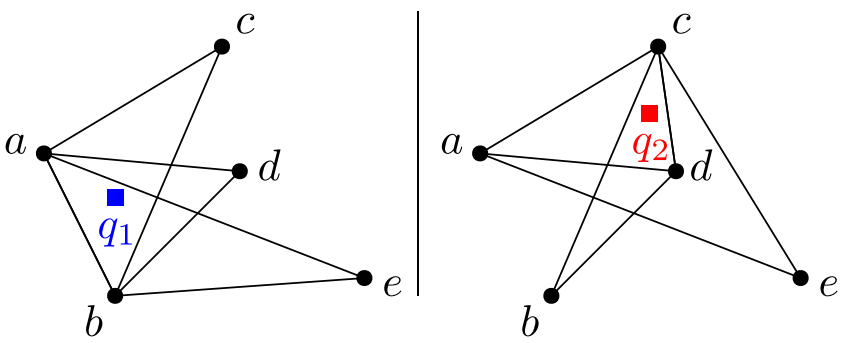


Fig. 1 Two examples of simplicial depth in the plane

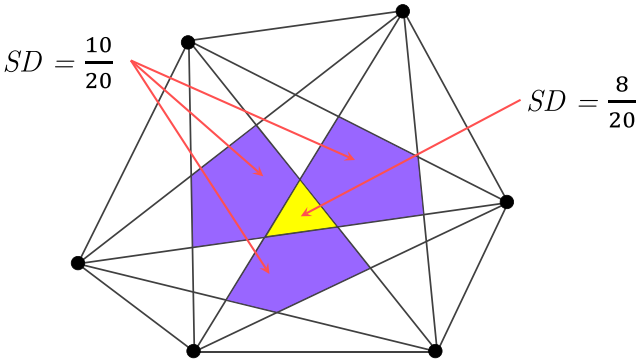


Fig. 2 Simplicial depth contours

$O(n^4)$ time [10]. In the higher dimension d, no better algorithm is known than the brute force method with $\Theta(n^{d+1})$ time.

3.2 Homomorphic encryption techniques

Homomorphic Encryption will effectively encourage the data owners to migrate their computation to cloud not only by reducing capital/operational costs, but also by enhancing privacy protection. It therefore adequately provides the opportunity to both store and process encrypted data. It can be broadly categorized into partially, somewhat, and fully forms with respect to the number of acceptable and supported operations. Recall that partially homomorphic encryption schemes support only either addition or multiplication function, whereas somewhat homomorphic schemes accept limited number of both aforesaid operations. Finally, fully homomorphic encryption schemes will tolerate arbitrary number of operations (i.e. both multiplication and addition). For example, RSA [33] and El-Gamal [11] are multiplicative, and Paillier [31] is an additive partially homomorphic schemes. However, BGN [5] is a somewhat homomorphic encryption that supports both addition and multiplication with the subtle difference that accepts arbitrary number of additions and only one multiplication.

Each HE scheme will be identified by four operations: KeyGen, Enc, Dec, and Eval function which performs the function f over ciphertext arguments. Moreover, they can utilize identical symmetric or distinct asymmetric keys for data encryption and ciphertext decryption. In symmetric mode both data owner and computational server are initiated by same keys whereas in asymmetric mode the private key must be kept absolutely private by data owner. Conversely, the public key can be safely disseminated to cloud servers.

During the recent decade, HE has received intense attention after the emergence of Fully Homomorphic Encryption. It led to a major breakthrough in practical computation offloading, where the proposed FHE schemes will be realistically able to carry out any desirable operations on ciphertexts. In the decade since Gentry's groundbreaking paper [14], the follow-up studies can be broken down into four main approaches: Ideal Lattice-based [14, 39], over Integers [42], (Ring)LWE-based (Learning with Error) [7], and NTRU-like [26]. Although the later schemes have improved the efficiency and performance of the implementations, the overhead and cost of the FHE implementations are highly application dependent. As an example, in [32], it is discussed that RLWE-based FHE has higher efficiency with a stronger security proof than LWE, but neither scheme is suitable for calculations involving division.

In order to encrypt the dataset, we utilize the recently implemented optimized BFV leveled-FHE scheme [36]. The BFV homomorphic scheme (based on Ring-LWE) comes originally from Brakerski's LWE-based fully homomorphic scheme which has been extended to Ring-LWE by Fan and Vercauteren [13]. This extension will empower the original scheme to realize fast and effective computation. Among different encryption schemes in the literature, two Ring-LWE-based leveled homomorphic encryption schemes, namely BFV and YASHE [6] are compared in [21]. Although YASHE is faster than BFV in some aspects, we prefer to use BFV in this study. The reason behind this preference is a distinguishable property of BFV which has practically and theoretically smaller noise growth during the homomorphic computation. In particular, BFV has been chosen because its noise growth is slower in the presence of multiplication operations (frequently used in simplicial depth computations). We briefly recall the BFV Ring-LWE based homomorphic encryption which will serve as our underlying building block.

- Parameter Generation: Given the security parameter λ, corresponding output $(d, \kappa, t, \chi_{key}, \chi_{err}, w)$ will be obtained, where:
  - ✓ Polynomial modulus d is a positive integer that defines ring $R = \mathbb{Z}[x]/(\Phi_d(x))$, in which $\Phi_d(x) \in \mathbb{Z}[x]$ is the d-th irreducible cyclotomic polynomial.
  - ✓ coefficient modulus κ and plaintext modulus t satisfy $1 < t < \kappa$.
  - ✓ Two discrete and bounded probability distributions χ_{key} and χ_{err} are chosen based on R.
  - ✓ $w > 1$ as an integer is the base of logarithm in $\ell = \lfloor log_w \kappa \rfloor$.
- Key Generation: The private, public, and relinearisation keys of the scheme are generated as follows:
  - ✓ Private key SK $= s$ can be obtained by sampling on χ_{key}.

✓ Public key $\mathsf{PK} = (b, a)$, where $b = [-(as + e)]_\kappa$ in which a and e are two uniformly random samples as: $a \leftarrow R_\kappa$ and $e \leftarrow \chi_{err}$.

✓ Relinearisation key $\mathsf{RelinK} = \zeta = ([w^i \cdot s^2 - (e_i + a_i \cdot s)]_\kappa, a_i) \in R^\ell$, where $a_i \leftarrow R_\kappa$ and $e_i \leftarrow \chi_{err}$ for $i \in [0, 1, ..., \ell]$.

– Encryption: Given public key $\mathsf{PK} = (b, a)$, ciphertext $\mathbf{c}$ for plaintext message m in message space R/tR can be calculated as

$$\mathbf{c} = (c_0, c_1) = \left(\left[\Delta\,[m]_t + bu + e_1\right]_\kappa, [au + e_2]_\kappa\right) \in R^2,$$

where e_1, e_2 are two samples from χ_{err}, u is a sample from χ_{key}, and $\Delta = \lfloor \kappa/t \rfloor$.

– Decryption: Given the ciphertext $\mathbf{c} = (c_0, c_1)$, the corresponding message $m = \left[\lfloor \frac{t}{\kappa} \cdot [c_0 + c_1 s]_\kappa \rceil\right]_t \in R$ can be recovered by the private key $\mathsf{SK} = s$.

– Addition ($\oplus$): Given two ciphertexts $\mathbf{c} = (c_0, c_1)$ and $\mathbf{c}' = (c_0', c_1')$, $c_\oplus = ([c_0 + c_0']_\kappa, [c_1 + c_1']_\kappa)$.

– Subtraction ($\ominus$): Given two ciphertexts $\mathbf{c} = (c_0, c_1)$ and $\mathbf{c}' = (c_0', c_1')$, $c_\ominus = ([c_0 - c_0']_\kappa, [c_1 - c_1']_\kappa)$.

– Multiplication ($\otimes$): Given two ciphertexts $\mathbf{c} = (c_0, c_1)$, $\mathbf{c}' = (c_0', c_1')$, and relinearisation key $\mathsf{RelinK} = \zeta = (\zeta_0, \zeta_1)$,

$c_\otimes = \mathsf{Relinearization}(\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \zeta)$

$= \left((\mathcal{C}_0 + \sum_{i=0}^{\ell} \zeta_{0i} \cdot \mathcal{C}_{2i}), (\mathcal{C}_1 + \sum_{i=0}^{\ell} \zeta_{1i} \cdot \mathcal{C}_{2i})\right)$, where

$$\mathcal{C}_0 = \left[\lfloor \frac{t}{\kappa} \cdot c_0 \cdot c_0' \rceil\right]_\kappa$$

$$\mathcal{C}_1 = \left[\lfloor \frac{t}{\kappa} \cdot (c_0 \cdot c_1' + c_1 \cdot c_0') \rfloor\right]_\kappa$$

$$\mathcal{C}_2 = \left[\lfloor \frac{t}{\kappa} \cdot c_1 \cdot c_1' \rceil\right]_\kappa$$

4 Models and design goal

In this section, we describe our system model, security model, related assumptions, and finally identify our design goal.

4.1 System model

The schematic in Fig. 3 depicts the overall system architecture. The main configuration components to compute the privacy-preserving simplicial depth of a query point are described next.

– **Dataset Owner:** This entity is a data asset owner who wants to perform basic operations to manipulate data. The dataset owner can specify the data dissemination process when the data is stored on the cloud. Typically the key generation phase will be initiated by dataset owner, and subsequent key storage and distribution steps should be considered in this phase. Meanwhile, different encrypted scaled versions of the dataset would be uploaded to adapt the proposed system to support floating point data.

– **Client Users:** They submit encrypted query points $q \in \mathbb{R}^d$ to the query processing server in order to obtain the simplicial depth value. Before submitting the query point, the client users encrypt each number using the public key (PK) which has been made available by the dataset owner. On the other hand, after receiving queries from client users, the query processing server is responsible for running Algorithm 1 to generate intermediate batch results. These batch values will be sent to the query response server where the final results will be obtained.

– **Collaborative Cloud-side Servers:** To securely offload simplicial depth computation over the cloud,

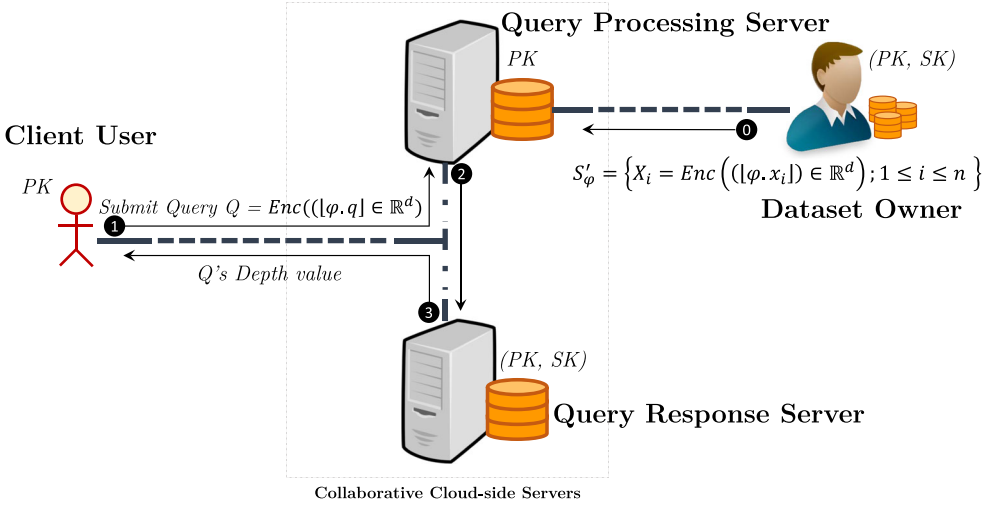


Fig. 3 System model of the privacy-preserving simplicial depth query

two cloud side servers (denoted by the dotted rectangle in Fig. 3) have been considered in our proposed scheme. Both the query processing server and the query response server are resource-abundant components that will work collaboratively to support our proposed $PSDQ$ scheme. They are honest-but-curious servers, which follow a collusion-free protocol in which any extra requests or communication will be ignored. The detailed description of each participant is provided below.

- **Query Processing Server:** This server acts as a mediator between data owner, client user, and the query response server. It performs *preprocessing steps* (i.e. computing $Dets$ and $DetsSq$) on any given encrypted dataset before persisting the dataset in the local data store. On the other side, when it receives the user's query, it performs Algorithm 1 to generate intermediate values that are processed by the query response server to finally deliver the simplicial depth value to the user.
- **Query Response Server:** For any $(d+1)$-set in the intermediate batch values, this server is responsible to determine whether the query point is an interior point of the corresponding closed simplices. To make such decision, it should have access to both public and private keys. Eventually, the normalized aggregated result will be sent back to the client user.

4.2 Security model

In our security model, both query processing and response servers are assumed to be honest-but-curious participants; i.e. the two follow the protocol but may try to extract additional information in the process. For example, cloud-side servers may be curious about user's queries or dataset values and the query response server may be curious about intermediate results received from the query processing server. Note that the honest-but-curious assumption would be responsible in practice since the cloud providers should protect their own reputation and financial interests. In addition, there would be no collusion between collaborative cloud-side servers. Otherwise, the private key can be revealed and exploited by adversaries to decrypt the dataset or queries. Since complying with collusion-free protocol has been supposed, the query processing server will contaminate the data before submitting the batch intermediate encrypted values to the query response server. Data contamination is performed by applying positive noise factors. Therefore, the privacy of the calculated batch result will be preserved and any sensitive information will not be guessable by query response server. Finally, it should be mentioned that to mitigate external active attacks, some mature message authentication code techniques or digital signature schemes could be applied. Although more efforts are usually required to implement them, nevertheless in this paper, we focus on the privacy-preserving simplicial depth query and consider those potential attacks beyond the scope of this paper.

4.3 Design goal

Our design goal is to propose a privacy-preserving simplicial depth query scheme over cloud to address the challenges in the above system model and security model. More specifically, in this paper, the following objectives should be realized.

- *Dataset owner's offloaded dataset and user submitted query should be protected.* In the proposed scheme, the encrypted dataset S'_{φ}, prepared and uploaded by the dataset owner, should be effectively secured against unauthorized access and disclosure, i.e. cloud-side servers and the client user cannot determine the plaintext dataset values. Moreover, the query point should also be protected, i.e. no one, except the query user, can determine the q or access the plaintext value of q.
- *The query processing and response value should be secured.* In the proposed $PSDQ$ scheme, not only the dataset and user's queries but also the computation over encrypted data should be protected. Particularly, the user requires to compute $Q \cdot \varphi$ and submit $Enc(Q \cdot \varphi)$ and φ to get the query response. Therefore, it would be impossible for both cloud-side servers to elicit any relevant information about user query. On the other hand, the query processing server has access to the encrypted dataset and it is responsible for generating encrypted intermediate contaminated batch values for further processing by query response server which will deliver the final depth value to user. Therefore, cloud-side servers will collaborate to each other to respond the client user without inferring the original dataset, query values or even intermediate results.
- *The proposed $PSDQ$ scheme should be efficient.* In order to achieve the above privacy requirements, additional computation costs will be incurred. Consequently, in the proposed $PSDQ$ scheme, we aim to reduce the response time by considering some *preprocessing steps* on ciphertext values, after uploading the encrypted dataset to the data store.

5 Our proposed *PSDQ* scheme

In this section, we will explore the $PSDQ$ scheme which mainly consists of the following elements: system initialization, client user query, generating intermediate batch results, and computing the simplicial depth value.

5.1 System initialization

In this phase, the dataset owner will setup the encryption method by considering security parameters to generate the public and private keys (PK) and (SK), respectively. Then, the dataset owner publishes his public key PK, and securely distributes the private key SK to the query response server. After that, the following protocol steps will be performed in dataset owner and cloud servers.

- *Step 1*: In order to securely offloading computation to the cloud, the dataset owner will encrypt each point of the dataset. Then, the encrypted data will be submitted to query processing server to be stored. Assuming the dataset S and scale factor $\varphi = \varphi(k) = 10^k$, where $k = 0, \cdots, \max\{|\text{decimal places}|\}$, then the scaled encrypted dataset S'_φ would be defined as follows:

$$\begin{aligned} &\forall\, x_i \qquad\qquad = (x_{i1}, x_{i2}, \ldots, x_{id}) \in S; \\ &X_i = (X_{i1}, X_{i2}, \ldots, X_{id}) \in S'_\varphi \;;\; X_{ij} = Enc(\lfloor \varphi \cdot x_{ij} \rfloor), \end{aligned}$$

- *Step 2*: For the computation of simplicial depth, to check whether the query point is an interior point of a given triangle, it is needed to compute 3 two-by-two determinants. Hence, to shorten the response time, after receiving the encrypted data, the query processing server will perform some preprocessing activities to generate and store the determinant of each triple encrypted data points. The corresponding results ($Dets$ and $DetsSq$), which are the proper inputs for Algorithm 1, would be stored in the local data store. In order to compute the elements of $Dets$, the function $DetCompute$ is required to be defined as follows. For every triple of encrypted planar points A, B, and C,

$$\begin{aligned} DetCompute(A, B, C) &= det(B \ominus A, C \ominus A) \\ &= det\left(\begin{bmatrix} B_1 \ominus A_1 & C_1 \ominus A_1 \\ B_2 \ominus A_2 & C_2 \ominus A_2) \end{bmatrix}\right) \\ &= ((B_1 \ominus A_1) \otimes (C_2 \ominus A_2)) \ominus \\ &\quad ((B_2 \ominus A_2) \otimes (C_1 \ominus A_1)). \end{aligned}$$

Referring to the definition, $Dets$ and $DetsSq$ can be given by:

$$Dets = \{DetCompute(X_i, X_j, X_k)\} \tag{2}$$

$$DetsSq = Dets \circledast Dets = \{D \otimes D;\ D \in Dets\}, \tag{3}$$

where $1 \le i < j < k \le n$ and $X_i, X_j, X_k \in S'_\varphi$.

5.2 Client user query

Given the public key, the end user will encrypt the query point $q = (q_1, q_2, \ldots, q_d) \in \mathbb{R}^d$, as follows:

$$\begin{aligned} Enc(\varphi \cdot q) &= (Enc(\varphi \cdot q_1), Enc(\varphi \cdot q_2), \ldots, Enc(\varphi \cdot q_d)) \\ &= (Q_1, Q_2, \ldots, Q_d) = Q. \end{aligned}$$

The obtained Q and the corresponding φ should be submitted to the query processing server in order to compute the simplicial depth value.

5.3 Generating intermediate batch results

After receiving the client user's encrypted query Q, processing server runs Algorithm 1 in order to generate the encrypted contaminated batch results of *barycentric coordinates* $\$ = \{\{\alpha, \beta, \gamma\}_t\}$. These results will be passed to the query response server.

Algorithm 1 Intermediate batch results.

Input: Q, S'_φ, Precomputed $Dets$ and $DetsSq$
Output: Encrypted batch $\$ = \left\{\{\alpha, \beta, \gamma\}_t; 1 \le t \le \binom{n}{3}\right\}$

1: $\$ \leftarrow \emptyset$
2: **for** each $X_i, X_j, X_k \in S'_\varphi$ **do**
3: $\quad D_{ijk} \leftarrow [Dets]_{ijk}$
4: $\quad \alpha' \leftarrow DetCompute(X_i, Q, X_k) \otimes D_{ijk}$
5: $\quad \beta' \leftarrow DetCompute(X_i, Q, X_j) \otimes D_{ijk}$
6: $\quad \gamma' \leftarrow [DetsSq]_{ijk} \ominus \alpha' \oplus \beta'$
7: $\quad (R_1, R_2, R_3) \leftarrow (Enc(r_1), Enc(r_2), Enc(r_3))$;
r_1, r_2, r_3 are positive integer random values
8: $\quad \{\alpha, \beta, \gamma\}_t \leftarrow (R_1 \otimes \alpha', R_2 \otimes \beta', R_3 \otimes \gamma')$
9: $\quad \$ \leftarrow \$ \cup \{\alpha, \beta, \gamma\}_t$
10: **return** $\$$

Note: To prevent the query response server from reconstructing the dataset or query point, each encrypted barycentric coordinate should be altered before inserting into the encrypted batch $. This alternation could be achieved by shuffling or contaminating the barycentric coordinates. In our implementation, to encumber the query response server from guessing not only the dataset and query but also the original α, β, and γ values, the contaminating method (line 7 of Algorithm 1) has been employed.

5.4 Computing simplicial depth value

Upon receiving the batch results $, the query response server decrypts each row of the batch results $\{\alpha, \beta, \gamma\} \in \$$ to check whether the current triple contributes the simplicial depth value. Finally, the aggregated value should be normalized by

the batch size, and reported to the client user. This process could be done by applying Algorithm 2.

Algorithm 2 Response depth value.

Input: Encrypted batch $\$ = \left\{\{\alpha, \beta, \gamma\}_t; 1 \leq t \leq \binom{n}{3}\right\}$
Output: Normalized $DepthValue$

1: $DepthValue \leftarrow 0$
2: **for** each $\{\alpha, \beta, \gamma\} \in \$$ **do**
3: $(A, B, \Gamma) \leftarrow (Dec(\alpha), Dec(\beta), Dec(\gamma))$
4: **if** ($A \geq 0$ *and* $B \geq 0$ *and* $\Gamma \geq 0$)
5: $DepthValue \leftarrow DepthValue + 1$
6: **return** $DepthValue/|\$|$

6 Security analysis

In this section, we analyze the security of the proposed $PSDQ$ scheme. More particularly, the privacy properties of encrypted dataset (S'_φ), encrypted scaled query point (Q) , and the reported normalized depth value are studied.

- *The encrypted dataset (S'_φ) is privacy-preserving*. In the proposed scheme, the floating point values are supported by offering different scale factors. The scaled dataset is encrypted using BFV scheme, and stored on external cloud server. As BFV has obtained its hardness from Ring Learning with Error (RLWE) problem, no one can access to original dataset value, without knowing the private key. Meanwhile, since we have assumed collusion-free protocol between cloud-side participants, the response query server won't also falsify or tamper the original dataset even accessing the private key.
- *The query Q and its related computation are also privacy preserving*. Using the public key PK, the client user can prepare and submit $Q = Enc(\varphi.q)$ to the cloud. Same as the client user, the query processing server has only access to the public key; hence, it would not be able to infer the user original query. Consequently, both the dataset and the user query would be hidden. Therefore, up to this point, the privacy preserving requirements on the simplicial depth query can be accomplished in $PSDQ$. Since BFV scheme can homomorphically evaluate the related arithmetic circuit, the query processing server performs Algorithm 1 to securely generate the encrypted batch results \$.
- *Generating the user response is privacy preserving*. The encrypted batch result should be forwarded to the query response server. Since the batch results are contaminated by applying some positive random noise (r_1, r_2, r_3), the query response server would not be able to predict original query point or dataset even if it has access to the private key. It will only decrypt the altered intermediate batch values to check whether they contributes in the $DepthValue$. In this step, the only required computation is comparing the decrypted values with zero. The decrypted positive values cause the $DepthValue$ to be incremented by one to represent the contribution of each entry in |\$| (Algorithm 2).

From the above analysis, our $PSDQ$ is arguably privacy preserving as a method for simplicial depth queries.

7 Experimental evaluation

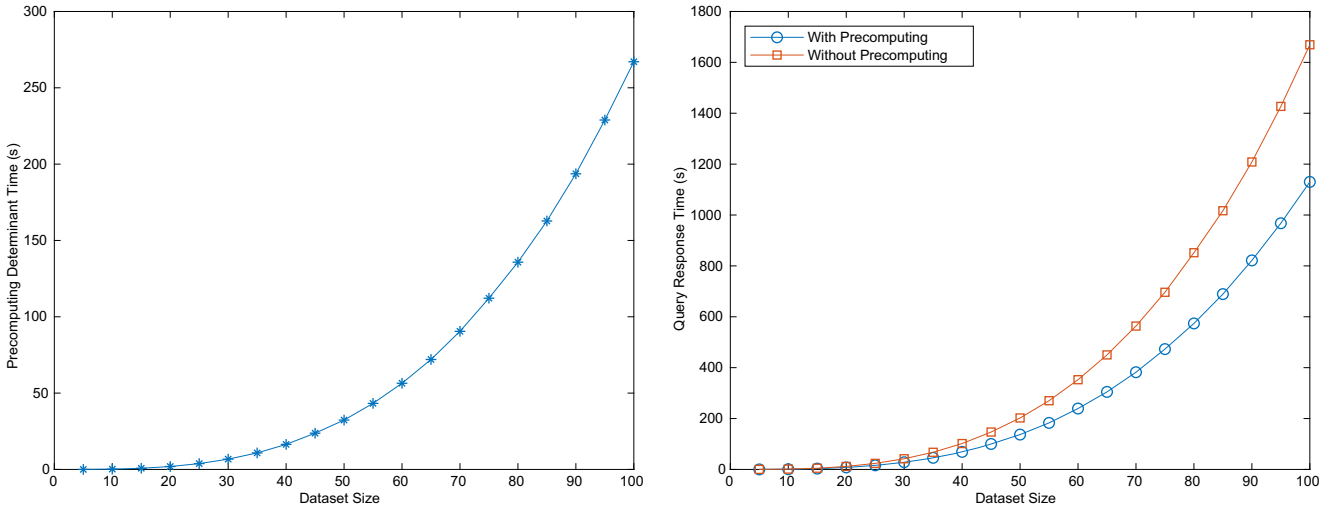
To evaluate the performance of the proposed $PSDQ$ scheme, we implemented both plaintext and ciphertext algorithms for computing planar simplicial depth of a query point. For ciphertext mode two different scenarios, with and without precomputation, are considered. We run our implementations on datasets with different sizes, i.e. n. The elements of both datasets and queries are some randomly generated points with four decimal places within the square $\{(x, y)|\ x, y \in [-1, 1]\}$. All the implementations are done in C++ together with SEAL library [36]. The experiments are conducted on a desktop computer with Intel i5-2400 3.1 GHz processor, 8 GB RAM, Ubuntu 16.04 platform. The detailed parameter setting are provided in Table 1.

7.1 Query response time

Generally speaking, the homomorphic computation over ciphertext in arithmetic circuits is a time consuming process. Since computing the simplicial depth requires large numbers of arithmetic operations, one way to decrease the query response time is to do all precomputable activities in advance. For example, in simplicial depth query, some of the two by two determinants (i.e. $Dets$ and $DetsSq$ in Eqs. 2 and 3) can be calculated and stored once the data is uploaded by the dataset owner. As illustrated in Fig. 4a, the required time for computing $Dets$ and $DetsSq$

Table 1 The parameter settings

Parameter	Value
Security parameter λ	$\lambda = 128$
Coeffient modulus $\kappa = \kappa_1.\kappa_2$	$\|\kappa_1\| = 55, \|\kappa_2\| = 56$
Polynomial modulus $\Phi_d = x^d + 1$	$d = 4096$
Plaintext modulus t	$t = 256$
Dataset size n	$n \in \{5, 10, 15, ..., 100\}$
Scale factor φ	$\varphi \in \{10, 100, 1000, 10000\}$



(a) Precomputing $Dets$ and $DetsSq$ (b) Query response time

Fig. 4 Computational costs in the proposed $PSDQ$ scheme

increases dramatically as the size of dataset increases. The results of our experiments for 10 queries are recorded. Figure 4b represents the differences in average response time of 10 queries for both scenarios (i.e. with and without precomputing). Especially for large datasets, the query processing time substantially reduces when $Dets$ and $DetsSq$ are precomputed.

7.2 Accuracy of the results

Plaintext modulus t for $PSDQ$ establishes the size of the plaintext data type. At the same time, it affects the noise budget determination and noise consumption in homomorphic computations. Therefore, trying to keep the plaintext data type as small as possible is essential to achieve the best computational performance. Since the proposed $PSDQ$ scheme accepts only integer values, floating point data should be converted into integer values using various scaling factors φ to form S'_φ. Consequently, a trade-off between desired accuracy and scaling factors will exist. To analyze this trade-off, two dissimilarity measures $|.,.|_e$ and $|.,.|_c$ are employed. These two measures respectively represent the data depth Euclidean distance and poset dissimilarity metric [37]. For given depth value vectors SD_a and SD_b, these measures are defined as:

$$|SD_a, SD_b|_e = 1 - r^2 \tag{4}$$

$$|SD_a, SD_b|_c = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} |M_{ij}^{SD_a} - M_{ij}^{SD_b}|}{n^2 - n}, \tag{5}$$

where r^2 is the coefficient of determination and matrix $M_{n\times n}^D$ is the Hamming matrix related to depth function D, and it is defined as follows:

$$M_{ij}^D = \begin{cases} 1 & depth_D(x_i) \le depth_D(x_j) \\ 0 & \text{otherwise.} \end{cases}$$

Since $r^2 \in [0, 1]$, the value of both Eqs. 4 and 5 are within $[0, 1]$, where the smaller value represents more similarity between SD_a and SD_b. The accuracy of the results can be computed in two following forms.

$$Acc_1 = 100\left(1 - \sqrt{|SD_a, SD_b|_e \cdot |SD_a, SD_b|_c}\right)$$

$$Acc_2 = 100\left(1 - \frac{|SD_a, SD_b|_e + |SD_a, SD_b|_c}{2}\right),$$

Table 2 A comparison between dissimilarity values

S'_φ	$\lvert(SD, SD_{PSDQ})\rvert_e$	$\lvert(SD, SD_{PSDQ})\rvert_c$	Acc_1	Acc_2
$\varphi = 10$	0.0359	0.0763	94.77	94.39
$\varphi = 100$	1.6860e-04	0.0116	99.86	99.41
$\varphi = 1000$	8.1835e-06	0.0012	99.99	99.94
$\varphi = 10000$	0	0	100	100

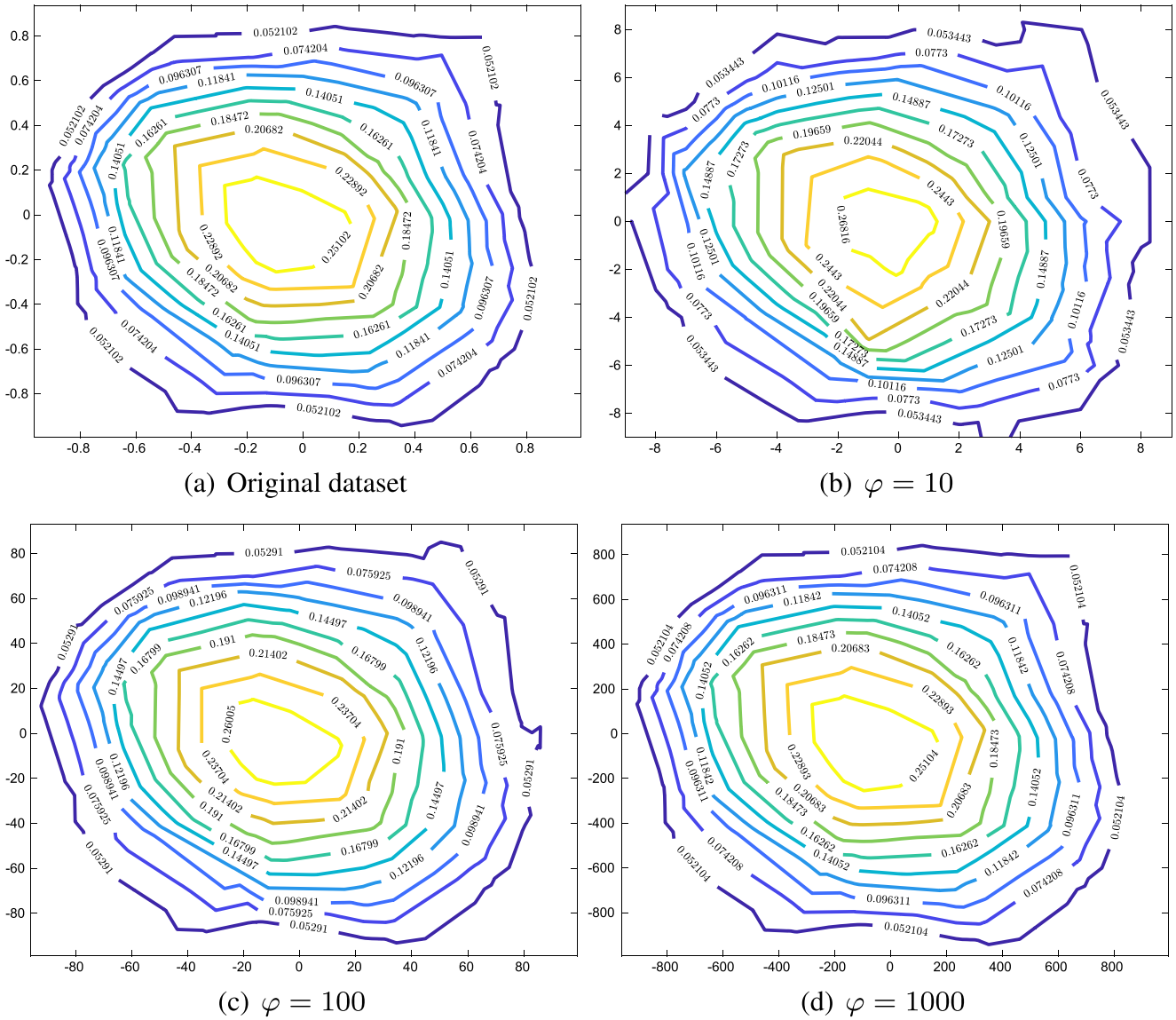


Fig. 5 Comparison of corresponding contours for different scale factors φ

where Acc_1 and Acc_2 are respectively defined based on the geometric mean and the arithmetic mean of two dissimilarity values.

Table 2 illustrates the accuracy comparison for SD and SD_{PSDQ} in simplicial depth computing with respect to different scaled encrypted datasets S'_{φ}.

In addition to the aforementioned results for different scalling factors φ, the corresponding contour plots of simplicial depth can be seen in Fig. 5b, c and d. For the scaling factor $\varphi = 10000$, the contour plot is ignored because the results for this case and for the original dataset with floating points values (Fig. 5a) are exactly the same (see the last row of Table 2).

8 Conclusion and future work

Although simplicial depth is one of the most influential and well-studied depth functions in both non-parametric data analysis and computational geometry, expensive to compute, especially in higher dimensions. In this paper, we presented a privacy preserving query scheme namely $PSDQ$ to compute the simplicial depth of a query point. In other words, considering the privacy-preserving goals, computing the simplicial depth of a query point $q \in \mathbb{R}^2$ with respect to a dataset $S \subset \mathbb{R}^2$, is outsourced. To do this, BFV RLWE-based homomorphic encryption scheme has been considered in order to offload both storing and computing

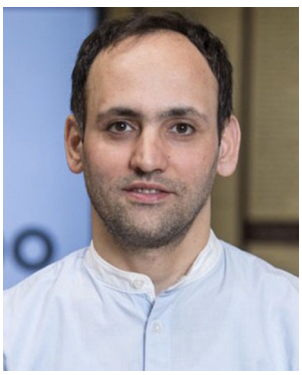
with the encrypted dataset. To support the proposed scheme, a series of experiments has been provided. The experimental results evaluate the performance and correctness of the developed scheme. Our proposed approach is general enough and independent from the dataset. Hence, the scheme also works on real-world datasets. In our future work, an extended scheme will be developed as a framework to outsource computing different depth functions over cloud servers.

References

1. Acar A, Aksu H, Selcuk Uluagac A., Conti M (2018) A survey on homomorphic encryption schemes: Theory and implementation. ACM Comput Surv 51:79:1–79:35
2. Aloupis G On computing geometric estimators of location, 2001. Master's thesis, M. Sc. McGill University
3. Aloupis G, Cortés C, Gómez F, Soss M, Toussaint G (2002) Lower bounds for computing statistical depth. Comput Stat Data Anal 40(2):223–229
4. Aloupis G, Langerman S, Soss M, Toussaint G (2003) Algorithms for bivariate medians and a fermat–torricelli problem for lines. Comput Geom 26(1):69–79
5. Boneh D, Goh E-J, Nissim K (2005) Evaluating 2-dnf formulas on ciphertexts. In: Theory of Cryptography Conference, Springer, pp 325–341
6. Bos JW, Lauter K, Loftus J, Naehrig M (2013) Improved security for a ring-based fully homomorphic encryption scheme. In: IMA International Conference on Cryptography and Coding, Springer, pp 45–64
7. Brakerski Z, Gentry C, Vaikuntanathan V (2014) (leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory (TOCT) 6(3):13
8. Chen D (2013) Algorithms for data depth. Carleton University
9. Chen ZQ (1995) Bounds for the breakdown point of the simplicial median. J Multivar Anal 55(1):1–13
10. Cheng AY, Ouyang M (2001) On algorithms for simplicial depth. In: CCCG, pp 53–56
11. ElGamal T (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory 31(4):469–472
12. Eppstein D, Goodrich MT, Tamassia R (2010) Privacy-preserving data-oblivious geometric algorithms for geographic data. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, pp 13–22
13. Fan J, Vercauteren F (2012) Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive 2012:144
14. Gentry C, Boneh D (2009) A fully homomorphic encryption scheme, vol 20. Stanford University Stanford
15. Hans S, Addepalli SC, Gupta A, Srinathan K (2009) On privacy preserving convex hull. In: International Conference on Availability, Reliability and Security, 2009. ARES'09. IEEE, pp 187–192
16. Hazewinkel M (2013) Encyclopaedia of Mathematics: C An updated and annotated translation of the Soviet ?Mathematical Encyclopaedia?, vol 2. Springer Science & Business Media
17. He X, Wang G (1997) Convergence of depth contours for multivariate datasets. The Annals of Statistics, pp 495–504
18. Hotelling H (1990) Stability in competition. In: The Collected Economics Articles of Harold Hotelling, Springer, pp 50–63
19. Jo S, Han J (2018) Convergence p2p cloud computing. Peer-to-Peer Networking and Applications 11(6):1153–1155
20. Langerman S, Steiger W (2000) The complexity of hyperplane depth in the plane. In: Symposium on Discrete Algorithms, ACM and SIAM
21. Lepoint T, Naehrig M (2014) A comparison of the homomorphic encryption schemes fv and yashe. In: International Conference on Cryptology in Africa, Springer, pp 318–335
22. Liang K, Yang B, He D, Zhou M (2011) Privacy-preserving computational geometry problems on conic sections. J Comput Inf Syst 7(6):1910–1923
23. Liu RY (1990) On a notion of data depth based on random simplices. The Annals of Statistics. pp 405–414
24. Liu RY (1995) Control charts for multivariate processes. J Am Stat Assoc 90(432):1380–1387
25. Liu RY, Serfling RJ, Souvaine DL (2006) Data depth: robust multivariate analysis, computational geometry, and applications, volume 72 American Mathematical Soc.
26. López-Alt A, Tromer E, Vaikuntanathan V (2012) On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the forty-fourth Annual ACM Symposium on Theory of Computing, ACM, pp 1219–1234
27. Lu W, Kawasaki S, Sakuma J Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data (this is the full version of the conference paper presented at ndss 2017). Technical report, IACR Cryptology ePrint Archive, Report 2016/1163 (2016). https://eprint.iacr.org/2016/1163.pdf
28. Miller K, Ramaswami S, Rousseeuw P, Sellarès T, Souvaine D, Streinu I, Struyf A (2001) Fast implementation of depth contours using topological sweep. In: Proceedings of the Twelfth annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, pp 690–699
29. Mosler K (2013) Depth statistics. In: Robustness and complex data structures, Springer, pp 17–34
30. Oja H (1983) Descriptive statistics for multivariate distributions. Statist Probab Lett 1(6):327–332
31. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, pp 223–238
32. Regev O (2010) The learning with errors problem. Invited survey in CCC, p 7
33. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126
34. Roman R, Lopez J, Mambo M (2018) Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Futur Gener Comput Syst 78:680–698
35. Rousseeuw PJ, Hubert M (1999) Regression depth. J Am Stat Assoc 94(446):388–402
36. Simple Encrypted Arithmetic Library (release 3.1.0) https://github.com/microsoft/SEAL, December 2018. Microsoft Research, Redmond, WA
37. Shahsavarifar R, Bremner D (2018) Approximate data depth revisited. arXiv:1805.07373
38. Small CG (1990) A survey of multidimensional medians. International Statistical Review/Revue Internationale de Statistique, pp 263–277
39. Smart NP, Vercauteren F (2010) Fully homomorphic encryption with relatively small key and ciphertext sizes. In: International Workshop on Public Key Cryptography, Springer, pp 420–443
40. Smith A (2011) Privacy-preserving statistical estimation with optimal convergence rates. In: Proceedings of the Forty-third

Annual ACM Symposium on Theory of Computing, ACM, pp 813–822
41. Tukey JW (1975) Mathematics and the picturing of data. In: Proceedings of the International Congress of Mathematicians, vol 2, pp 523–531
42. Dijk MartenVan, Gentry Craig, Halevi Shai, Vaikuntanathan Vinod (2010) Fully homomorphic encryption over the integers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, pp 24–43
43. Wu D, Haven J (2012) Using homomorphic encryption for large scale statistical analysis. Technical report Technical report: cs. stanford. edu/people/dwu4/papers/FHESI Report pdf
44. Zuo Y, Serfling R (2000) General notions of statistical depth function. Annals of Statistics, pp 461–482

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Hassan Mahdikhani received the B.Eng.degree in Computer Engineering- Software from Kharazmi University, Tehran, Iran, in 2001 and the M.Eng. degree in Computer Engineering-Software from Iran University of Science and Technology, Tehran, Iran, in 2006. He is currently a Ph.D. candidate in Computer Science at the University of New Brunswick (UNB), and a cybersecurity researcher at the Canadian Institute for Cybersecurity (CIC), Canada. His research interests include secure and privacy-preserving computation offloading and applied cryptography.

Rasoul Shahsavarifar holds a PhD degree in computer science from University of New Brunswick (UNB), Canada. Rasoul is currently doing postdoctoral fellow in the faculty of Computer Science at UNB under the supervision of Dr. David Bremner. Rasoul's research areas include Algorithm design and analysis, Data depth, computational geometry, and multivariate data analysis. The title of his PhD thesis is: "algorithmic and geometric aspects of data depth with focus on beta-skeleton depth".

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

David Bremner holds three degrees in Computer Science, a B.Sc. Hons. from the University of Calgary (1990), an M.Sc. from Simon Fraser University (1993) and a Ph.D. from McGill University (1997). David spent two years as an NSERC postdoctoral fellow in the Department of Mathematics at the University of Washington from 1997 to 1999. Since 2000 David has been a faculty member at the University of New Brunswick, and is currently a Professor of Computer Science with a Cross Appointment to the Department of Mathematics and Statistics. Recently, David has made extended research visits to the Technical University of Berlin (as an Alexander von Humboldt Fellow), the University of Rostok, and Kyoto University.