Circular Range Search on Encrypted Spatial Data

Boyang Wang†, Ming Li†, Haitao Wang† and Hui Li§
† Department of Computer Science, Utah State University, Logan, UT, USA
§ State Key Laboratory of Integrated Service Networks, Xidian University, China

***Abstract*—Searchable encryption is a promising technique enabling meaningful search operations to be performed on encrypted databases while protecting user privacy from untrusted third-party service providers. However, while most of the existing works focus on common SQL queries, geometric queries on encrypted spatial data have not been well studied. Especially, circular range search is an important type of geometric query on spatial data which has wide applications, such as proximity testing in Location-Based Services and Delaunay triangulation in computational geometry.**

In this paper, we propose two novel symmetric-key searchable encryption schemes supporting circular range search. Informally, both of our schemes can correctly verify whether a point is inside a circle on encrypted spatial data without revealing data privacy or query privacy to a semi-honest cloud server. We formally define the security of our proposed schemes, prove that they are secure under Selective Chosen-Plaintext Attacks, and evaluate their performance through experiments in a real-world cloud platform (Amazon EC2). To the best of our knowledge, this paper represents the first study in secure circular range search on encrypted spatial data.

I. Introduction

Motivation. Geometric range query is an important and common type of query in spatial databases, where spatial locations are represented as data points in a Euclidean space and queries can be described as geometric objects such as rectangles, circles, etc. Especially, **circular range search** aims at finding data points (which could be multi-dimensional) that are located inside a circular range, as shown in Fig. 1. It has wide applications in geographic information systems, computer-aid design and computational geometry [1].

For example, in Location-Based Service (LBS), a user can perform *proximity testing* [2] (e.g., finding friends, restaurants, WiFi hotspots within a certain distance based on his/her current location) by running circular range search on a geographic map; similarly, a Mac-OS device can detect another Mac-OS device within approximately 30 feet (9 meters) with an application named AirDrop in order to easily share large data files; a mobile user can leverage the feature of *friend radar* in WeChat, which is an instant message app with 438 million active users as August 2014, to locate other WeChat users that are close to him/her by carrying out circular range search; in computational geometry, verifying whether a triangulation T of a point set S is a Delaunay triangulation can be done by performing circular range search to see if any point from S is inside any circumcircle of a triangulation of T [3].

On the other hand, with the emergence of cloud computing, more and more companies/service providers begin to outsource their large datasets to third-party cloud service providers (CSPs), such as Google and Amazon, to reduce their local cost

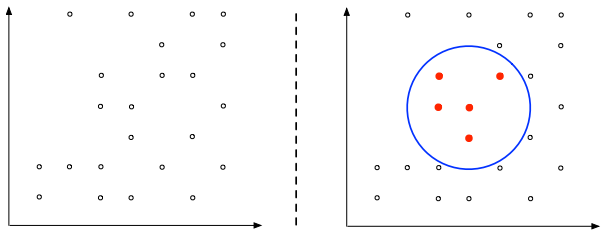

Fig. 1. A set of points in a plane, and a circular range query finds the points that inside a circle.

on data management and data storage. However, this raises serious privacy concerns from the users of those companies because the CSPs may not be fully trusted in handling their data, and there have been numerous data breach incidents in the past [4]. Foursquare, a well-known LBS company outsources its location database containing all their users' geographic location information to Amazon Web Services (AWS). Meanwhile, it is common for a Foursquare user to run proximity-based friend finding over her own friend list (all stored in the cloud). Considering geographic location information are sensitive to not only users but also to the company (due to legal and commercial issues), in this example both *data privacy* (i.e., the locations of users in the database) and *query privacy* (i.e., the location of the querier) should be protected from AWS. Another example where preserving privacy against a third-party CSP is needed, is a hospital that outsources the database of its patients' private locations to a public cloud. A doctor from this hospital may need to query those spatial data in order to locate and monitor the patients that are close to him/her within a certain range in order to provide emergency healthcare.

A direct approach to relief users' concerns on the privacy leakage of outsourced spatial data is to let the company encrypt its database with traditional encryption (e.g., using AES). Unfortunately, it makes the realization of any data search functionality a challenging task. As a result, many searchable encryption schemes [4]–[12] have been introduced to allow users to perform meaningful search on encrypted data without revealing *data privacy* nor *query privacy* to the cloud server.

However, most of the existing studies in searchable encryption [4]–[12] only focus on common SQL queries, such as Boolean keyword search, which are not suitable for supporting geometric range queries [1] on spatial data. Although the extensions of comparison search in several recent solutions [13]–[17] can support rectangular range search (i.e., retrieving points that are inside a rectangle), none of the existing searchable encryption schemes has particularly studied circular range search.

Our Contributions. In this paper, we propose two searchable encryption schemes particularly supporting circular range queries on encrypted spatial data. With our proposed schemes, a semi-honest cloud server is able to correctly verify whether a point is inside a circle in the ciphertext domain. Informally speaking, except knowing the *Boolean* search result of a circular range query over encrypted spatial data (*i.e., inside or outside a circle*) and *radius pattern* (*i.e., the radius of each circular range query*), the cloud server is not able to reveal additional privacy about data or queries. Our main contributions are summarized as follows:

1) We first design a symmetric-key Circle Predicate Encryption (CPE) based on an existing Predicate Encryption [18] (named as SSW in this paper). This CPE is able to predicate *whether a point is on the boundary of a circle* on encrypted spatial data, and it will be leveraged as a stepping stone for our later design in circular range search.
2) We formally describe the definition and security of symmetric-key Circular Range Searchable Encryption (CRSE), and propose two novel CRSE schemes (named CRSE-I and CRSE-II respectively) to support circular range search on encrypted spatial data. The main difference between these two proposed schemes is that the first one is stronger in terms of privacy protection, while the second one is much efficient and scalable. To the best of our knowledge, our work represents the very first study in secure circular range search on encrypted spatial data.
3) We prove our proposed schemes are secure under Selective Chosen-Plaintext Attacks (SCPA) [19] in terms of data privacy and query privacy. In addition, we conduct experiments in Amazon EC2 to evaluate the performance of our schemes in real cloud platform.

II. Related Work

In this section, we particularly describe some research that are closely related to circular range search on encrypted data.

Rectangular Range Search. Rectangular range search, which retrieves all the points inside a rectangle, is an alternative approach to conduct circular range search, even in the plaintext domain (*e.g., generating a minimal rectangle that covers all the area of a circle*) [1]. However, this alternative introduces many false positives, where the false positives indicate the points that are inside the minimal rectangle but are not inside the original circle.

Multi-dimensional (conjunctive) range searchable encryption schemes [13]–[17] essentially provide solutions supporting rectangular range search. Specifically, both Boneh et al. [13] and Shi et al. [14] designed public-key schemes which are able to handle rectangular range queries with linear search. Moving a step forward, several schemes [15]–[17] proposed to achieve faster-than-linear rectangular range search by utilizing tree structures, such as R-trees [16], [17] or kd-trees [15]. Recent research [11], [12] supporting range queries can also be extended to operate rectangular range search on encrypted data. In addition, simply using Order-Preserving Encryption [20], [21] with multiple dimensions is also another option to enable rectangular range search on encrypted spatial data.

Nearest Neighbor Search. Nearest neighbor search or k-nearest neighbor search is to find the nearest point (or k-nearest points) for a given query point by evaluating Euclidean distance. Since the query point and the results of nearest neighbor search can also form a circle (*where the query point is the center, and the distance between the query point and the nearest neighbor is the radius*), it seems nearest neighbor search is similar as circular range search.

However, the main difference is that nearest neighbor search pre-defines the number of *effective* search results (i.e., 1 or k) in the generation of each query without providing a particular radius; while circular range search needs to specifically define a radius in each query (before search) without considering the number of effective search results. Therefore, nearest neighbor search is different from circular range search, even in the plaintext domain.

Nearest neighbor search or k-nearest neighbor search over encrypted data was first studied in [22], which achieves linear search and is vulnerable under Chosen-Plaintext Attacks. Its subsequent works focus on improving search efficiency [23] by using data structures or minimizing client-server interactions [24] with two non-colluding servers.

Private Proximity Testing. Some recent works [2], [25], [26] considering private proximity testing directly between two users are also related to circular range search over encrypted spatial data we study in this paper. Specifically, with private proximity testing, a user Alice is able to test whether another user Bob is within some certain distance without revealing any user's specific location to each other.

In these schemes [2], [25], [26], the private computation and comparison of the distance between Alice and Bob are generally achieved by using Secure Two-Party Computation [27] (e.g., Garbled Circuits [26]), which inevitably introduce *multiple rounds of interactions* between the two parties. While our work is targeting a design with a *minimal* one-round client-server interaction (see some further discussions in the next section). Besides, the traditional two-user setting (or even with an additional use of an *oblivious* computation server [2]) of these schemes is different from our model. Specifically, our model is a standard searchable encryption model, which involves a cloud server storing encrypted datasets but necessarily revealing Boolean search results (i.e., this cloud server in our model is not required to be *totally oblivious*).

III. Problem Statement

System Model. Our system model in Fig. 2 is a standard searchable encryption model, which includes a data owner, a data user and the cloud server. A data owner (e.g., a company) stores its dataset in the cloud server to reduce local cost on data storage and hardware. A data user (e.g., a user of the company) would like to search over outsourced spatial data in the cloud. The cloud server provides data storage and search services.

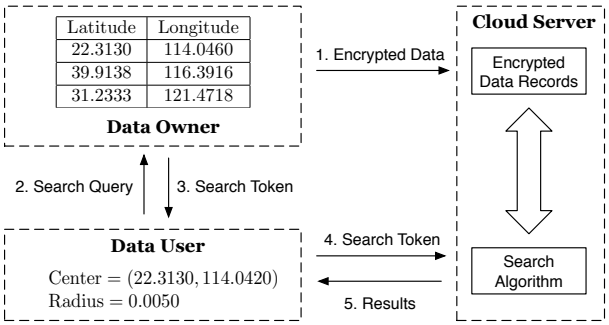

Fig. 2. The system model with a data owner, a data user and the cloud server

Note that the data owner itself always has the capability to search over outsourced spatial data.

In this study, we focus on circular range search, which means each data record in the outsourced spatial dataset can be represented as a point while a search query is a circle. A circle is defined by its center and radius. The purpose of circular range search is to find points that are inside a given query (as described in Fig. 1).

Due to privacy concern (e.g., legal and commercial issues), the data owner encrypts its dataset and only outsources the encrypted dataset to the cloud server; and a client (i.e., a data user or the data owner) only submits each search query's encrypted form (i.e., a search token) to the cloud server. The data owner manages the secret keys for encrypting data and generating search tokens. A data user trusts the data owner, but not the cloud server. The cloud server is *semi-honest* (i.e., *honest-but-curious*) [4], which means it provides reliable data and search services, but it is curious about the content of data records outsourced by the data owner and the content of circular range queries submitted by a client.

A Straightforward Design. The essential problem of solving circular range search is to evaluate whether a data point is inside a query circle. Mapping to the general logic in the plaintext domain, a straightforward design for circular range search on encrypted spatial data is to **compute** the distance between two points (i.e., a data point and the center of the query circle), and **then compare** this distance with the radius of the query circle in the ciphertext domain. Although current (efficient) cryptographic primitives can evaluate the above two operations in the ciphertext domain independently (e.g. Additively Homomorphic Encryption [28] for computation and Order-Preserving Encryption [20], [21] for comparison), they are not able to evaluate these two operations *continuously* in the ciphertext domain. As a result, the straightforward design with this **compute-then-compare** logic on encrypted spatial data will inevitably introduce heavy interactions between a client and the cloud server or the impractical assumption of two (or more) non-colluding servers. These limitations also exist in the current searchable encryption schemes supporting nearest neighbor search [23], [24], which requires a similar compute-then-compare process on encrypted data.

In order to avoid these limitations, our design presented later in this paper is able to support circular range search without using the compute-then-compare logic. As a result, a client is able to correctly finish circular range search on encrypted spatial data with a (**minimal**) one-round client-server interaction (*i.e., a client submits a search token and the cloud server returns search results*) using a **single** cloud server. As a necessary trade-off, our design will leak radius pattern, which is only a minor leakage (see Sec. IV).

Notations. Before we introduce the formal definition of a symmetric-key Circular Range Searchable Encryption (CRSE), we first clarify some notations.

First, we assume the values of points and the centers of circles in this paper are all integers[1]. We use Δ_T^w to denote the *data space*, where w is the number of dimensions and T is the size of each dimension. Each element in data space Δ_T^w is essentially a w-dimensional point, where the value of this point in each dimension is within $[0, T-1]$ (without loss of generality). For the ease of description, we first assume $w = 2$ and the size of each dimension is the same (i.e., $T_k = T$, for $1 \leq k \leq w$). We will discuss the extension supporting higher dimensions (i.e., $w > 2$) in Sec. VI.

Essentially, a data record D in the preceding system model is a point and a circular range query Q is a circle, where $D \in \Delta_T^w$ and $Q \subseteq \Delta_T^w$. We present a circular range query as $Q = \{(x_c, y_c), R\}$, where (x_c, y_c) is the center and R is the radius of the circle. In the following definition of symmetric-key CRSE, we use $D \in Q$ to denote point D is inside[2] circle Q, and use $D \notin Q$ to describe point D is outside circle Q.

***Definition** 1:* **(Symmetric-Key CRSE).** *A symmetric-key CRSE is a tuple of four polynomial-time algorithms* $\Pi = (\texttt{GenKey}, \texttt{Enc}, \texttt{GenToken}, \texttt{Search})$ *such that:*

- $\texttt{SK} \leftarrow \texttt{GenKey}(1^\lambda, \Delta_T^w)$*: is a probabilistic key generation algorithm that is run by the data owner to setup the scheme. It takes as input a security parameter* λ *and the data space* Δ_T^w*, and outputs a secret key* SK*.*
- $C \leftarrow \texttt{Enc}(\texttt{SK}, D)$*: is a probabilistic algorithm run by the data owner to encrypt a data record. It takes as input a secret key* SK *and a data record* D*, where* $D \in \Delta_T^w$*, and outputs a ciphertext* C*.*
- $\texttt{TK} \leftarrow \texttt{GenToken}(\texttt{SK}, Q)$*: is a probabilistic algorithm run by the data owner to generate a search token for a given circular range query. It takes as input a secret key* SK *and a circular range query* $Q = \{(x_c, y_c), R\}$*, where* $Q \subseteq \Delta_T^w$*, and outputs a search token* TK*.*
- $I \leftarrow \texttt{Search}(\texttt{TK}, C)$*: is a deterministic algorithm run by the server to search on encrypted data. It takes as input a search token* TK *and a ciphertext* C*, and returns an identifier* I *(e.g., a memory location in the cloud server) of ciphertext* C*, if the corresponding data record* $D \in Q$*; otherwise, outputs* $\perp$*.*

Correctness. We say that the above symmetric-key CRSE scheme is correct if for all $\lambda \in \mathbb{N}$, all SK output by

[1]Or floating numbers with fixed digits, such as the ones in Fig. 2, that can be smoothly converted to integers. The reasons that why floating numbers with unfixed digits are not supported in our design are explained in Sec. VI.

[2]For the ease of description, when we mention a point is inside a circle in the rest of this paper, we indicate that this point is geometrically inside the boundary of this circle or it is exactly on the boundary of this circle.

$\texttt{GenKey}(1^\lambda, \Delta_T^w)$, all $D \in \Delta_T^w$, all C output by $\texttt{Enc}(\texttt{SK}, D)$, all $Q \subseteq \Delta_T^w$, all TK output by $\texttt{GenToken}(\texttt{SK}, Q)$,

- If $D \in Q$: $\texttt{Search}(\texttt{TK}, C) = I$;
- If $D \notin Q$: $\Pr[\texttt{Search}(\texttt{TK}, C) = \perp] \geq 1 - \texttt{negl}(\lambda)$;

where $\texttt{negl}(\lambda)$ denotes a *negligible function* [29] in λ.

Informally, the above correctness indicates that algorithm Search will return the identifier of ciphertext C, if its corresponding point D is indeed inside circle Q; otherwise, algorithm Search will return the identifier of this ciphertext with only a negligible probability. The encryption and decryption of the content of each data record itself can always be independently performed with another layer of traditional encryption. There is no need to specifically describe it in the definition of a searchable encryption according to the previous study [13].

The simplest description of the preceding searchable encryption on the entire dataset $\mathbf{D} = \{D_1, ..., D_n\}$ can be easily extended by separately encrypting each D_i with $\texttt{Enc}(\texttt{SK}, D_i)$ and linearly searching each ciphertext C_i with $\texttt{Search}(\texttt{TK}, C_i)$ (if assuming no additional data structures is leveraged).

IV. Security Definition

Leakage Function. Before we introduce the security definitions of a symmetric-key CRSE, we first present a *leakage function* $\mathcal{L}$, which covers all the information leakage in a symmetric-key CRSE. For instance, the privacy leakage introduced by a query Q on a data record D is denoted as $\mathcal{L}(D, Q)$. Informally, the privacy leakage function of a symmetric-key CRSE induced by circular range queries on a set of data records includes the following aspects:

- ***Size Pattern****: The cloud server learns the total number of data records in the dataset and the total number of search queries submitted by a client.*
- ***Access Pattern****: The cloud server reveals the identifier of each encrypted data record that are returned for each query.*
- ***Search Pattern****: The cloud server learns if the same encrypted data record is retrieved by two different queries.*
- ***Radius Pattern****: The cloud server knows the radius of a circular range query submitted by a client.*

Note that size pattern, access pattern and search pattern are general information leakage in searchable encryption [6]. Theoretically, using Oblivious RAM [30] can preserve access pattern and search pattern. However, it is not efficient in practice. How to preserve the information defined in the above leakage function is out of the scope of this paper.

Radius pattern are particularly introduced for circular range queries we study in this work. Note that radius pattern is essentially a *part* of information in a circular range query. The query privacy we claim to protect in this paper particularly focuses on *preserving the values of centers (i.e., x_c and y_c)* against the semi-honest cloud server. And we argue that revealing radius pattern is only a minor leakage in circular range search compared to revealing the values of centers. For example, revealing a query circle's radius is 100 meters in Location-Based Services will not directly indicate a user's private location (i.e., the center of the query circle). Because the center of the query circle could still be anywhere in the plane even with the knowledge of the radius. In fact, the leakage of radius pattern in our design is similar as the leakage of the message length in traditional encryption (such as AES). The reason that why radius pattern are leaked and an additional approach to *somewhat* preserve radius pattern in practice are both described in Sec. VI.

Security Definition of CRSE. The security objective of a symmetry-key CRSE is to preserve *data privacy* and *query privacy*, which can be rigorously formalized with *indistinguishability* [29] under Selective Chosen-Plaintext Attacks (SCPA) [19] by following the security of some previous searchable encryptions [18], [31].

Query Privacy. Informally, *query privacy* of a symmetric-key CRSE under Selective Chosen-Plaintext Attacks means by submitting two circular range queries Q_0 and Q_1 with a same radius R, a computationally bounded adversary $\mathcal{A}$ is able to adaptively issue a number of ciphertext requests and token requests *selectively* restricted by Q_0, Q_1 and leakage function $\mathcal{L}$. However, this adversary is not able to (computationally) distinguish between Q_0 and Q_1. See the rigorous definition (Def. 2) in Appendix.

Data Privacy. Informally, *data privacy* of a symmetric-key CRSE under Selective Chosen-Plaintext Attacks means that by submitting two data records D_0 and D_1, a computationally bounded adversary $\mathcal{A}$ is able to adaptively issue a number of ciphertext requests and token requests *selectively* restricted by D_0, D_1 and leakage function $\mathcal{L}$. However, adversary $\mathcal{A}$ fails to distinguish between D_0 and D_1. See the rigorous definition (Def. 3) in Appendix.

V. Circle Predicate Encryption

In this section, we first introduce a Predicate Encryption [18] (denoted as SSW) as a primitive, and then design a Circle Predicate Encryption (CPE) to test *whether a point is on the boundary of a circle* on encrypted spatial data based on SSW. This CPE will be used as a stepping stone for the design of Circular Range Searchable Encryption in the next section.

Predicate Encryption. Predicate Encryption is a special type of encryption, which is able to test whether plain data (e.g., u) satisfies a predicate (i.e., $f(u) = 1$ or $f(u) = 0$) without revealing the content of plain data. Shen, Shi and Waters (SSW) [18] designed a symmetric-key Predicate Encryption, and it is able to support *inner product* queries. Specifically, data is described as a vector $\vec{u}$ and a predicate can be denoted as a vector $\vec{v}$, and the evaluation over encrypted data reveals $f(\vec{u}) = 1$ iff $\vec{u} \circ \vec{v} = 0$, where $\vec{u} \circ \vec{v} = \sum_{i=1}^{n} v_i \cdot u_i$ denotes the inner product of the two vectors. Besides protecting data privacy, SSW can preserve query privacy as well under Selective Chosen-Plaintext Attacks. A brief description of each algorithm in SSW is presented in Fig. 3. Further discussions and security analyses of SSW can be found in [18].

Circle Predicate Encryption. Interestingly, we can leverage SSW to build a Circle Predicate Encryption, which is able

- $\texttt{Setup}(1^\lambda, \Delta_T^w)$: Given security parameter λ and data space Δ_T^w, output secret key $\texttt{SK}$.
- $\texttt{Enc}(\texttt{SK}, \vec{u})$: Given $\texttt{SK}$ and plaintext $\vec{u} \in \Delta_T^w$, where $\vec{u} = (u_1, ..., u_w)$, output ciphertext C.
- $\texttt{GenToken}(\texttt{SK}, \vec{v})$: Given $\texttt{SK}$ and query $\vec{v} \in \Delta_T^w$, where $\vec{v} = (v_1, ..., v_w)$, output token $\texttt{TK}$.
- $\texttt{Query}(\texttt{TK}, C)$: Given $\texttt{TK}$ and C, output 1 iff $\vec{u} \circ \vec{v} = 0$ and output 0 otherwise.

Correctness: SSW is correct, for all λ, all $\vec{u} \in \Delta_T^w$, all $\vec{v} \in \Delta_T^w$, all $\texttt{SK} \leftarrow \texttt{Setup}(1^\lambda, \Delta_T^w)$, all $C \leftarrow \texttt{Enc}(\texttt{SK}, \vec{u})$, all $\texttt{TK} \leftarrow \texttt{GenToken}(\texttt{SK}, \vec{v})$,

- If $\vec{u} \circ \vec{v} = 0$, $\texttt{Query}(\texttt{TK}, C) = 1$;
- If $\vec{u} \circ \vec{v} \neq 0$, $\Pr[\texttt{Query}(\texttt{TK}, C) = 0] \geq 1 - \texttt{negl}(\lambda)$;

Fig. 3. Details of SSW.

to particularly evaluate whether a point is on the boundary of a circle on encrypted data. The main idea here is to "split" the equation of a circle, where this equation is essentially a *polynomial*, into an expression of an inner product of two vectors. More specifically, we can easily have a polynomial

$$P = (x - x_c)^2 + (y - y_c)^2 - R^2 = 0, \tag{1}$$

where (x_c, y_c) is the center and R is the radius of a circle Q and (x, y) denotes any point that is on the boundary of this circle. Then, we simply split this polynomial into an expression of an inner product of two vectors:

$$\begin{aligned} & (x - x_c)^2 + (y - y_c)^2 - R^2 \\ = \ & (x^2 + y^2) - 2x \cdot x_c - 2y \cdot y_c + x_c^2 + y_c^2 - R^2 \\ = \ & (x^2 + y^2) \cdot 1 + (-2x) \cdot x_c + (-2y) \cdot y_c + 1 \cdot (x_c^2 + y_c^2 - R^2) \\ = \ & (x^2 + y^2, -2x, -2y, 1) \circ (1, x_c, y_c, x_c^2 + y_c^2 - R^2) \\ = \ & \textstyle\sum_{i=1}^4 u_i \cdot v_i, \end{aligned} \tag{2}$$

where $\vec{u} = (u_1, u_2, u_3, u_4) = (x^2 + y^2, -2x, -2y, 1)$ and $\vec{v} = (v_1, v_2, v_3, v_4) = (1, x_c, y_c, x_c^2 + y_c^2 - R^2)$, and we have

$$(x - x_c)^2 + (y - y_c)^2 - R^2 = 0 \quad \Leftrightarrow \quad \sum_{i=1}^4 u_i \cdot v_i = 0. \tag{3}$$

Of course, this polynomial P presented above can be denoted as an inner product of many different pairs of two vectors. The *key principle* of this splitting process is: separating the point (i.e., x and y) into vector $\vec{u}$ and distributing the circle (i.e., x_c, y_c and R) into another vector $\vec{v}$. We use an algorithm $\texttt{Split}(P)$ to denote this splitting process on polynomial P, where the output include the length α of vector $\vec{u}$ and $\vec{v}$, formula $f_{\vec{u}}$ and $f_{\vec{v}}$ describing the general forms of vector $\vec{u}$ and $\vec{v}$. Specifically, the corresponding output of $\texttt{Split}(P)$ in the above example are described as:

$$\begin{aligned} \alpha = w + 2, \quad f_{\vec{u}} = (x^2 + y^2, -2x, -2y, 1), \\ f_{\vec{v}} = (1, x_c, y_c, x_c^2 + y_c^2 - R^2). \end{aligned} \tag{4}$$

Clearly, assuming the number of dimensions w is given (*i.e., the general form of polynomial P is given*), formula $f_{\vec{u}}$ and $f_{\vec{v}}$ can be obtained from $\texttt{Split}$ without knowing any exact values of points or circles. Therefore, $\texttt{Split}$ is a *deterministic* algorithm.

- $\texttt{GenKey}(1^\lambda, \Delta_T^w)$: Given a security parameter λ and the data space Δ_T^w, calculate polynomial P, compute

$$\{\alpha, f_{\vec{u}}, f_{\vec{v}}\} \leftarrow \texttt{Split}(P), \quad \texttt{SK} \leftarrow \texttt{SSW.Setup}(1^\lambda, \Delta_T^\alpha),$$

  set $\{w, T, \alpha, f_{\vec{u}}, f_{\vec{v}}\}$ as public parameters and output a secret key $\texttt{SK}$.
- $\texttt{Enc}(\texttt{SK}, D)$: Given a secret key $\texttt{SK}$ and a point D, where $D \in \Delta_T^w$, compute

$$\vec{u} \leftarrow f_{\vec{u}}(D) \in \Delta_T^\alpha, \quad C \leftarrow \texttt{SSW.Enc}(\texttt{SK}, \vec{u}),$$

  and output a ciphertext C.
- $\texttt{GenToken}(\texttt{SK}, Q)$: Given a secret key $\texttt{SK}$ and a circle Q, where $Q \subseteq \Delta_T^w$, compute

$$\vec{v} \leftarrow f_{\vec{v}}(Q) \in \Delta_T^\alpha, \quad \texttt{TK} \leftarrow \texttt{SSW.GenToken}(\texttt{SK}, \vec{v}),$$

  and output a token $\texttt{TK}$.
- $\texttt{Query}(\texttt{TK}, C)$: Given a token $\texttt{TK}$ and a ciphertext C, compute $\texttt{Flag} \leftarrow \texttt{SSW.Query}(\texttt{TK}, C)$, where $\texttt{Flag} = 1$ iff $D \in^* Q$ and $\texttt{Flag} = 0$ otherwise, and output $\texttt{Flag}$.

Correctness of CPE. We say CPE is correct, if for all λ, all $D \in \Delta_T^w$, all $Q \subseteq \Delta_T^w$, all $\texttt{SK} \leftarrow \texttt{GenKey}(1^\lambda, \Delta_T^w)$, all $C \leftarrow \texttt{Enc}(\texttt{SK}, D)$, all $\texttt{TK} \leftarrow \texttt{GenToken}(\texttt{SK}, Q)$,

- If $D \in^* Q$, $\texttt{Query}(\texttt{TK}, C) = 1$;
- If $D \notin^* Q$, $\Pr[\texttt{Query}(\texttt{TK}, C) = 0] \geq 1 - \texttt{negl}(\lambda)$;

Fig. 4. Details of Circle Predicate Encryption.

With the above expression of an inner product on polynomial P, we can build a Circle Predicate Encryption (CPE), which is able to test whether a point $D = (x, y)$ is on the boundary of a circle $Q = \{(x_c, y_c), R\}$ on encrypted spatial data *without revealing the content of the point, the center or the radius*. In the rest of this paper, we use $D \in^* Q$ to indicate point D is on the boundary of circle Q; otherwise, we have $D \notin^* Q$. Details of CPE are illustrated in Fig. 4. Adapting to the cloud scenario, a data owner can run $\texttt{GenKey}$, $\texttt{Enc}$ and $\texttt{GenToken}$, then the cloud server is able to verify whether a point is on the boundary of a circle on encrypted spatial data by running $\texttt{Query}$ without revealing data or query privacy.

An Example for CPE. Here is a concrete example helping readers to understand how CPE works based on SSW. For instance, given the number of dimension $w = 2$, a point $D = (x, y) = (2, 2)$, a circle $Q = \{(x_c, y_c), R\} = \{(3, 2), 1\}$ as shown in Fig. 5. According to Eq. 4, a data owner first computes

$$\begin{aligned} \alpha = 4, \quad \vec{u} = f_{\vec{u}}(D) = (8, -4, -4, 1), \\ \vec{v} = f_{\vec{v}}(Q) = (1, 3, 2, 12). \end{aligned}$$

Then, it encrypts $\vec{u} = (8, -4, -4, 1)$ with $\texttt{SSW.Enc}(\texttt{SK}, \vec{u})$ to generate a ciphertext C, and calculates a search token $\texttt{TK}$ with $\texttt{SSW.GenToken}(\texttt{SK}, \vec{v})$, where $\vec{v} = (1, 3, 2, 12)$. Later, the cloud server is able to evaluate $\texttt{SSW.Query}(\texttt{TK}, C)$ and output $\texttt{Flag} = 1$, because

$$\begin{aligned} \vec{u} \circ \vec{v} \ &= \ (8, -4, -4, 1) \circ (1, 3, 2, 12) \\ &= \ 8 - 12 - 8 + 12 \ = \ 0 \end{aligned}$$

which verifies point D is on the boundary of circle Q. On the contrary, for another point $D' = (1, 3)$ in Fig. 5, its vector form is $\vec{u}' = f_{\vec{u}}(D') = (10, -2, -6, 1)$. The evaluation on

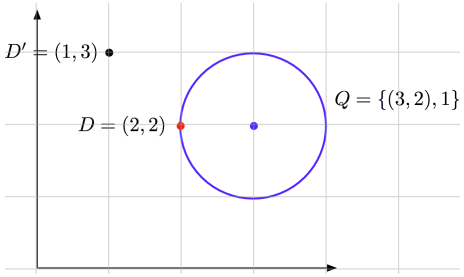

Fig. 5. A concrete example

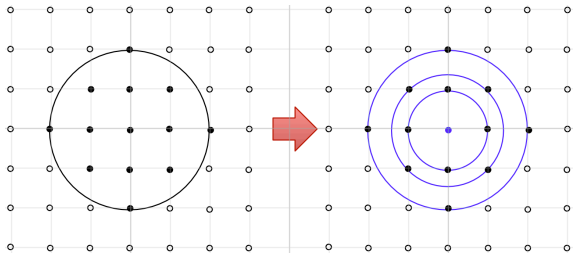

Fig. 6. The **Main Idea** of our design: given a circular range query, generate a number of concentric circles covering all the possible points inside.

$\texttt{SSW.Query}(\texttt{TK}, C')$ will output $\texttt{Flag} = 0$ (i.e., point D' is not on the boundary of circle Q), since $\vec{u}' \circ \vec{v} = (10, -2, -6, 1) \circ (1, 3, 2, 12) = 4 \neq 0$.

Security of CPE. Since CPE is essentially a special instance of SSW with pre-defined forms of $\vec{u}$ and $\vec{v}$ by `Split`. The correctness and security of CPE can be simply obtained from SSW. More concretely, because SSW is able to protect data privacy and query privacy, this CPE is clearly able to protect the privacy of vector $\vec{u}$ and $\vec{v}$. As a result, the privacy of a point, a center and a radius are all preserved during the evaluation of `Query` in CPE.

Cases with Higher Dimensions. Our design in CPE can also be simply extended to support points with higher dimensions. For instance, if $w = 3$, the polynomial of a *sphere* is

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = R^2,$$

where (x_c, y_c, z_c) is the center of this sphere Q, R is the radius and (x, y, z) represents any point that is on the boundary of this sphere. Similarly, we can use the same principle to split this polynomial into an inner product of two vectors with `Split`, where the output of it in this case are described as:

$$\alpha = w + 2, \quad f_{\vec{u}} = (x^2 + y^2 + z^2, -2x, -2y, -2z, 1),$$
$$f_{\vec{v}} = (1, x_c, y_c, z_c, x_c^2 + y_c^2 + z_c^2 - R^2).$$

VI. Circular Range Searchable Encryption

In this section, we use Circular Predicate Encryption presented above as a stepping stone to design two symmetric-key Circular Range Searchable Encryption schemes, denoted as CRSE-I and CRSE-II respectively. Specifically, CRSE-I is able to strictly achieve the security under Selective Chosen-Plaintext Attacks defined by Def. 2 and Def. 3 in Appendix, but it has some limitations in terms of scalability and efficiency. While CRSE-II is more scalable and efficient, but it is weaker in the aspect of security compared to the first one. The essential objective of each scheme is to verify *whether a point is inside a circle* on encrypted spatial data. In fact, the starting points of the design of these two schemes are the same, the main differences are introduced due to the different styles of implementing Circular Predicate Encryption.

A. The Starting Point of the Design

Main Idea. Basically, the main idea of this starting point in our design is to use a number of *concentric circles* (*i.e., the same center but different radiuses*) to cover all the possible points inside the given *query circle* Q (as shown in Fig. 6). Clearly, this query circle itself is the one with the largest radius among these concentric circles. For the center, we treat it as a special concentric circle with a radius of zero. The correctness of this idea lies in the fact that *if a point is on the boundary of one of these concentric circles, then it is certainly inside the given query circle; otherwise, it is outside the query circle*.

The Number of Concentric Circles. Because we only consider data with integers in this paper, so a *finite* number of concentric circles is sufficient to cover all the possible points. We denote this number as m. We use $\texttt{GenConCircle}(Q)$ to denote the algorithm of computing m and calculating all the radiuses of those concentric circles based on a query circle Q, where $Q = \{(x_c, y_c), R\}$. Since the output of `GenConCircle` is *independent* with the center, we can also present it equivalently as $\texttt{GenConCircle}(R)$.

Assume all the m radiuses of those concentric circles are $\{r_1, ..., r_m\}$, and we have $r_i^2 = x^2 + y^2$ (*since* `GenConCirlce` *is independent with the value of the center, we assume the current center is* $(0, 0)$ *without loss of generality*). Because the radius of each concentric circle is *distinct*, given the restriction $x^2 + y^2 = r_i^2 \leq R^2$ based on a query circle, the number of concentric circles is actually equivalent to the number of integers in $[0, R^2]$ that can be represented as **a sum of two squares**. While the sum of two squares, which is presented as below, is a classic problem in number theory [32].

***Theorem** 1:* **(Sum of Two Squares)** *A positive integer n is the sum of two squares if and only if each prime factor p of n such that $p \equiv 3 \mod 4$ occurs to an even power in the prime factorization of n.*

Detailed proof of this theorem can be found in [32]. With this knowledge, we can easily compute the actual value of m and enumerate all the radiuses of those concentric circles. Clearly, this `GenConCircle` is a *deterministic* algorithm and the upper bound of m is $R^2 + 1$ considering in two dimensions.

B. Circular Range Searchable Encryption-I

Now let us see how to specifically verify a point inside a circle on encrypted spatial data with CRSE-I. With m concentric circles we described in the preceding subsection, we can have m polynomials

$$P_i = (x - x_c) + (y - y_c) - r_i^2, \quad \textit{for } 1 \leq i \leq m$$

The idea of CRSE-I is to combine all the m polynomials into a *single polynomial* P, split this single polynomial P into two vectors, and run it as a special instance of SSW as a similar

$$
\begin{aligned}
P &= P_1 \times P_2 \\
&= \left[(x^2+y^2)+(-2x\cdot x_c)+(-2y\cdot y_c)+(x_c^2+y_c^2-r_1^2)\right]\cdot\left[(x^2+y^2)+(-2x\cdot x_c)+(-2y\cdot y_c)+(x_c^2+y_c^2-r_2^2)\right] \\
&= (x^2+y^2)^2\cdot 1 + (-2x^3-2xy^2)\cdot x_c + (-2x^2y-2y^3)\cdot y_c + (x^2+y^2)\cdot(x_c^2+y_c^2-r_2^2) \\
&\quad + (-2x^3-2xy^2)\cdot x_c + 4x^2\cdot x_c^2 + 4xy\cdot x_c y_c + (-2x)\cdot(x_c^3+x_c y_c^2-x_c r_2^2) + (-2x^2y-2y^3)\cdot y_c \\
&\quad + 4xy\cdot x_c y_c + 4y^2\cdot y_c^2 + (-2y)\cdot(x_c^2 y_c+y_c^3-y_c r_2^2) + (x^2+y^2)\cdot(x_c^2+y_c^2-r_1^2) \\
&\quad + (-2x)\cdot(x_c^3+x_c y_c^2-x_c r_1^2) + (-2y)\cdot(x_c^2 y_c+y_c^3-y_c r_1^2) + 1\cdot(x_c^2+y_c^2-r_1^2)(x_c^2+y_c^2-r_2^2) \\
&= [(x^2+y^2)^2,\ -2x^3-2xy^2,\ -2x^2y-2y^3,\ x^2+y^2,\ -2x^3-2xy^2,\ 4x^2,\ 4xy,\ -2x,\ -2x^2y-2y^3,\ 4xy, \\
&\quad 4y^2,\ -2y,\ x^2+y^2,\ -2x,\ -2y,\ 1]\circ[1,\ x_c,\ y_c,\ x_c^2+y_c^2-r_2^2,\ x_c,\ x_c^2,\ x_c y_c,\ x_c^3+x_c y_c^2-x_c r_2^2,\ y_c, \\
&\quad x_c y_c,\ y_c^2,\ x_c^2 y_c+y_c^3-y_c r_2^2,\ x_c^2+y_c^2-r_1^2,\ x_c^3+x_c y_c^2-x_c r_1^2,\ x_c^2 y_c+y_c^3-y_c r_1^2,\ (x_c^2+y_c^2-r_1^2)(x_c^2+y_c^2-r_2^2)] \\
&= f_{\vec{u}}\circ f_{\vec{v}} \qquad (5)
\end{aligned}
$$

way as CPE in the preceding section. More specifically, we have

$$P = P_1 \times P_2 \times \cdots \times P_m. \qquad (6)$$

In this case, if a point is on the boundary of any of those concentric circles, it is inside the circle, and vice verse. The correctness of this description can be presented as follows:

$$P = 0 \Leftrightarrow P_i = 0, \quad \textit{for any } i \in [1, m]. \qquad (7)$$

This single polynomial P can be easily computed from `GenConCircle`, which is essentially able to output all the m polynomials (i.e., P_i, for $1 \le i \le m$) of concentric circles. After calculating this single polynomial P, we can follow the same splitting principle in CPE to similarly divide polynomial P into an inner product of two vectors using `Split`. Specifically, according to the results in CPE, for each sub-polynomial P_i, we can split it with $\alpha = (w+2)$ *terms* (i.e., an inner product of two $(w+2)$-dimensional vectors).

$$
\begin{aligned}
P_i &= (x^2+y^2, -2x, -2y, 1)\circ(1, x_c, y_c, x_c^2+y_c^2-r_i^2) \\
&= \underbrace{(x^2+y^2)\cdot 1}_{term} + \underbrace{(-2x)\cdot x_c}_{term} + \underbrace{(-2y)\cdot y_c}_{term} + \underbrace{1\cdot(x_c^2+y_c^2-r_i^2)}_{term}
\end{aligned}
$$

Therefore, the splitting on the single polynomial P ends up with $\alpha = (w+2)^m$ terms (i.e., an inner product of two $(w+2)^m$-dimensional vectors). An example of the splitting on this single polynomial P where $m = 2$ is described in Eq. 5. Note that the vectors' length α can be reduced by further simplifying polynomial P (e.g., the optimized value of α could be 10 in Eq. 5 instead of $(w+2)^m = 4^2 = 16$).

Clearly, given the number of dimensions w and the number of concentric circles m (*i.e., the general form of polynomial P is given*), formula $f_{\vec{u}}$ and $f_{\vec{v}}$ can be obtained from `Split` without knowing any exact values of points or circles. However, since we need the pre-information about the radius R (i.e., the number of concentric circles m) in order to decide the general form of polynomial P, one *inherent* limitation of this CRSE-I in terms of scalability is that it can only handle circular range queries with *static radius* (i.e., the radius of circular range queries needs to be pre-defined during the setup the scheme). In addition, since the radius in this design needs to be a public parameter and `GenConCircle` and `Split` are both deterministic algorithms, which inevitably leak radius pattern of circular range queries. That is why we particularly include radius pattern in the leakage function. Details of CRSE-I are presented in Fig. 7. The correctness of CRSE-I can be easily obtained based on SSW similar as Circular Predicate Encryption.

- $\texttt{GenKey}(1^\lambda, \Delta_T^w)$: Given a security parameter λ, the data space Δ_T^w, define radius R, compute

  $$\{m, r_1, ..., r_m\} \leftarrow \texttt{GenConCircle}(R),$$

  calculate polynomial $P = P_1 \cdots P_m$, compute

  $$\{\alpha, f_{\vec{u}}, f_{\vec{v}}\} \leftarrow \texttt{Split}(P), \quad \texttt{SK} \leftarrow \texttt{SSW.Setup}(1^\lambda, \Delta_T^\alpha),$$

  set $\{w, T, R, \alpha, f_{\vec{u}}, f_{\vec{v}}\}$ as public parameters and output a secret key `SK`.
- $\texttt{Enc}(\texttt{SK}, D)$: Given a secret key `SK` and a point D, where $D = (x, y) \in \Delta_T^w$, compute

  $$\vec{u} \leftarrow f_{\vec{u}}(D) \in \Delta_T^\alpha, \quad C \leftarrow \texttt{SSW.Enc}(\texttt{SK}, \vec{u}),$$

  and output a ciphertext C.
- $\texttt{GenToken}(\texttt{SK}, Q)$: Given a secret key `SK` and a circle $Q = \{(x_c, y_c), R\}$, where $Q \subseteq \Delta_T^w$, compute

  $$\vec{v} \leftarrow f_{\vec{v}}(Q) \in \Delta_T^\alpha, \quad \texttt{TK} \leftarrow \texttt{SSW.Query}(\texttt{SK}, \vec{v}),$$

  and output a token `TK`.
- $\texttt{Search}(\texttt{TK}, C)$: Given a token `TK` and a ciphertext C, compute $\texttt{Flag} \leftarrow \texttt{SSW.Query}(\texttt{TK}, C)$, where $\texttt{Flag} \in \{0, 1\}$, if `Flag` equals to 1, return the identifier I of this ciphertext; otherwise output $\perp$.

Fig. 7. Details of CRSE-I.

An Example for CRSE-I. Here we still use the previous concrete example in Fig. 5 to show how CRSE-I works based on SSW. Specifically, given the number of dimensions $w = 2$, a circular range query $Q = \{(x_c, y_c), R\} = \{(3, 2), 1\}$, a point $D = (x, y) = (2, 2)$ from Fig. 5, we can first have the number of concentric circles m is 2, which is sufficient to cover all the five possible points inside circle Q in Fig. 5 based on the sum of two squares theorem. In addition, we have the radius of these two concentric circles are $r_1 = 0$ (i.e., representing the center as a concentric circle with radius zero) and $r_2 = 1$ (i.e., a concentric circle that is equivalent to the query circle itself) respectively. According to Eq. 5, a data owner can compute the vector form of point $D = (2, 2)$ as $\vec{u} = f_{\vec{u}}(D) =$

$$(64, -32, -32, 8, -32, 16, 16, -4, -32, 16, 16, -4, 8, -4, -4, 1)$$

And the vector form of circle $Q = \{(x_c, y_c), R\} = \{(3, 2), 1\}$ can be computed (based on Eq. 5 with the knowledge of $r_1 = 0$ and $r_2 = 1$) as $\vec{v} = f_{\vec{v}}(Q) =$

$$(1, 3, 2, 12, 3, 9, 6, 36, 2, 6, 4, 24, 13, 39, 26, 156)$$

Then, the data owner is able to output a ciphertext C with $\mathtt{SSW.Enc}(\mathtt{SK}, \vec{u})$, and can later compute a search token $\mathtt{TK}$ with $\mathtt{SSW.GenToken}(\mathtt{SK}, \vec{v})$. Then, the cloud server is able to evaluate $\mathtt{SSW.Query}(\mathtt{TK}, C)$ and output $\mathtt{Flag} = 1$, because $\vec{u} \circ \vec{v} = 0$, which verifies point $D = (2, 2)$ is inside circle $Q = \{(3, 2), 1\}$.

On the contrary, for point $D' = (1, 3)$ in Fig. 5, its vector form (based on Eq. 5) is $\vec{u}' = f_{\vec{u}}(D') =$

$$(100, -20, -60, 10, -20, 4, 12, -2, -60, 12, 36, -6, 10, -2, -6, 1)$$

The evaluation on $\mathtt{SSW.Query}(\mathtt{TK}, C')$ will output $\mathtt{Flag} = 0$ (i.e., point D' is not inside circle Q), since $\vec{u}' \circ \vec{v} = 20 \neq 0$.

Search Complexity of CRSE-I. According to the above design and search complexity of SSW [18], the search efficiency of CRSE-I is $O(\alpha^m)$ per data record. The number of concentric circles m increases with $O(R^2)$ based on the discussion on the sum of two squares theorem. Clearly, although CRSE-I is able to correctly verify whether a point is inside a circle on encrypted spatial data, it is impractical for circular range queries with large radiuses.

C. Circular Range Searchable Encryption-II

In order to overcome the limitations in terms of scalability and efficiency in the first scheme, we now design another one, denoted as CRSE-II, which is still based on the idea of concentric circles. Specifically, in this second scheme, instead of combing the m polynomials into a single one compared to CRSE-I, we directly split each sub-polynomial P_i into an inner product of two vectors (as we did in CPE), and create a sub-token for each concentric circle by leveraging CPE (essentially SSW). Since we have m concentric circles in total in order to cover all the possible points inside a query circle, CRSE-II will generate m sub-tokens for a query circle. Correspondingly, CRSE-II needs to evaluate a number of m sub-tokens on a ciphertext to test whether a point is inside the query circle in the *worst case* (i.e., if the evaluation of the i-th sub-token on a ciphertext outputs 1, where $i \in [1, m]$, the sub-tokens from $(i+1)$-th to m-th do not need to be evaluated anymore). Details of CRSE-II are presented in Fig. 8. Obviously, the correctness of CRSE-II is based on the correctness of CPE.

Since $\mathtt{GenConCircle}$ is a deterministic algorithm as we mentioned before, revealing the number of sub-tokens m in the output of $\mathtt{GenToken}$ in CRSE-II essentially reveals radius R. This is the reason that CRSE-II fails to protect radius privacy. Besides $\mathtt{GenConCircle}$, an additional permutation function $\mathtt{Permute}$ is used in CRSE-II. The purpose of this permutation is to permute the order of sub-tokens in $\mathbf{TK} = \{\mathtt{TK}_1, ..., \mathtt{TK}_m\}$ into a random order. Clearly, this permutation function needs to be permuted with a fresh random β each time to maintain its security.

An Example for CRSE-II. We continue to use the example in Fig. 5 to show how our second scheme works based on CPE (essentially SSW). Given the number of dimensions $w = 2$, a point $D = (2, 2)$ and a query circle $Q = \{(3, 2), 1\}$, we first have the number of concentric circles $m = 2$ with concentric circle radius $r_1 = 0$ and $r_2 = 1$ respectively (still based on the sum of two squares theorem as in our first scheme CRSE-I). According to Eq. 2 in CPE, we have

$$\begin{aligned} P_1 &= (x^2 + y^2, -2x, -2y, 1) \circ (1, x_c, y_c, x_c^2 + y_c^2 - r_1^2) \\ P_2 &= (x^2 + y^2, -2x, -2y, 1) \circ (1, x_c, y_c, x_c^2 + y_c^2 - r_2^2) \end{aligned}$$

which can be used to calculate the vector form of point D as $\vec{u} = (8, -4, -1, 1)$ (as the same as the one in CPE) and the vector forms of two concentric circles $Q_1 = \{(x_c, y_c), r_1\} = \{(3, 2), 0\}$, $Q_2 = \{(x_c, y_c), r_2\} = \{(3, 2,), 1\}$ as

$$\vec{v_1} = (1, 3, 2, 13), \qquad \vec{v_2} = (1, 3, 2, 12)$$

Then a data owner can generate a ciphertext C with $\mathtt{SSW.Enc}(\mathtt{SK}, \vec{u})$, compute $\mathbf{TK} = \{\mathtt{TK}_1, \mathtt{TK}_2\}$ with two sub-tokens where $\mathtt{TK}_1 = \mathtt{SSW.GenToken}(\mathtt{SK}, \vec{v_1})$ and $\mathtt{TK}_2 = \mathtt{SSW.GenToken}(\mathtt{SK}, \vec{v_2})$, and output a permuted search token as $\mathbf{TK}^* = \{\mathtt{TK}_1^*, \mathtt{TK}_2^*\} = \mathtt{Permute}(\mathbf{TK}, \beta)$.

Later, the cloud server is able to verify point $D = (2, 2)$ is inside circle $Q = \{(3, 2), 1\}$ on encrypted spatial data by running $\mathtt{SSW.Query}(\mathtt{TK}_i^*, C)$, for $1 \leq i \leq m$. It is because the evaluation on one of these two sub-tokens with ciphertext C outputs $\mathtt{Flag}_i = 1$ (i.e., one of the two inner products is zero, where $\vec{u} \circ \vec{v}_1 \neq 0$ but $\vec{u} \circ \vec{v}_2 = 0$).

On the contrary, for point $D' = (1, 3)$, its vector form is $\vec{u}' = (10, -2, -6, 1)$. The evaluation on its corresponding ciphertext C' with the search token $\mathbf{TK}^* = \{\mathtt{TK}_1^*, \mathtt{TK}_2^*\}$ of query circle $Q = \{(3, 2), 1\}$ will indicate this point is outside this circle. The reason is that the evaluation on both of the two sub-tokens with ciphertext C', where $\mathtt{SSW.Query}(\mathtt{TK}_i^*, C')$, for $1 \leq i \leq m$, will output $\mathtt{Flag}_i = 0$ (i.e., neither of the two inner products is zero, where $\vec{u}' \circ \vec{v}_1 \neq 0$ and $\vec{u}' \circ \vec{v}_2 \neq 0$).

Search Complexity of CRSE-II. The search complexity of CRSE-II on each ciphertext is $O(\alpha m)$ in the worst case (i.e., $O(\alpha)$ for each sub-token), where $\alpha = (w + 2)$ is the length

- $\mathtt{GenKey}(1^\lambda, \Delta_T^w)$: same as $\mathtt{CPE.GenKey}(1^\lambda, \Delta_T^w)$;
- $\mathtt{Enc}(\mathtt{SK}, D)$: same as $\mathtt{CPE.Enc}(\mathtt{SK}, D)$;
- $\mathtt{GenToken}(\mathtt{SK}, Q)$: Given a secret key $\mathtt{SK}$ and a circle Q, where $Q = \{(x_c, y_c), R\}$ and $Q \subseteq \Delta_T^w$, compute

  $$\{m, r_1, ..., r_m\} \leftarrow \mathtt{GenConCircle}(R),$$
  $$Q_i = \{(x_c, y_c), r_i\}, \quad \mathtt{TK}_i \leftarrow \mathtt{CPE.GenToken}(\mathtt{SK}, Q_i)$$

  for $1 \leq i \leq m$ and $\mathbf{TK} = (\mathtt{TK}_1, ..., \mathtt{TK}_m)$, generate a random $\beta \in [1, m!]$, calculate

  $$\mathbf{TK}^* \leftarrow \mathtt{Permute}(\mathbf{TK}, \beta),$$

  and output a token $\mathbf{TK}^* = (\mathtt{TK}_1^*, ..., \mathtt{TK}_m^*)$;
- $\mathtt{Search}(\mathbf{TK}^*, C)$: Given a token $\mathbf{TK}^* = (\mathtt{TK}_1^*, ..., \mathtt{TK}_m^*)$ and a ciphertext C, for $1 \leq i \leq m$, compute

  $$\mathtt{Flag}_i \leftarrow \mathtt{CPE.Query}(\mathtt{TK}_i^*, C),$$

  if any $\mathtt{Flag}_i$ equals to 1, return the identifier I of this ciphertext; otherwise output $\perp$.

Fig. 8. Details of CRSE-II.

of the vectors and w is the number of dimensions. Obviously, this $O(\alpha m)$ is much efficient than $O(\alpha^m)$ in CRSE-I.

D. Additional Discussions.

CRSE-I and CRSE-II in Multiple Dimensions. Since Circular Predicate Encryption is the stepping stone for both of the two schemes, and CPE can be easily extended to support spatial data in multiple dimensions as we described in Sec. V, therefore, both CRSE-I and CRSE-II can be extended to provide circular range search on encrypted spatial data with multiple dimensions. One major difference is that the number of concentric circles may increase.

Specifically, when it comes to higher dimensions, the number of concentric circles we need can be enumerated based on Legendre's theorem on the sum of three squares (i.e., 3 dimensions) and Lagrange's theorem on the sum of four squares (i.e., 4 dimensions) [32]. Particularly, when the number of dimensions is equal or larger than 4, the number of concentric circles is exactly $R^2 + 1$, because every integer in $[0, R^2]$ can be represented as a sum of four squares according to Lagrange's four-square theorem.

Floating Numbers. Note that we cannot support floating numbers (e.g., $\frac{1}{3}$ and $\sqrt{5}$) with our design methodology because the number of concentric circles could be infinite (i.e., the number of points need to be covered is infinite), which is much more challenging to handle on encrypted spatial data. Another main reason is that an encryption algorithm itself generally does not support floating numbers (i.e., values to be encrypted are normally elements of $\mathbb{Z}_p$, where p is a large prime). This second reason is also why other geometric searchable encryption schemes, such as for rectangle range search [13]–[16], limit their implementations to integers only.

It is worth to notice that, even we assume all the points and centers are integers, the radius R could be floating numbers in many cases. For instance, $R = \sqrt{2}$ (e.g., $(x, y) = (1, 1)$ and $(x_c, y_c) = (0, 0)$). However, since we are dealing with the square of a radius, e.g., $R^2 = 2$ (which is still an integer) during the encryption, so it is not a problem for our methods to handle cases like this.

Radius Privacy. As we mentioned, our schemes leak radius pattern, which is a minor leakage. However, we can still use an approach to somewhat preserve radius privacy in practice. Take CRSE-II as an example, we can introduce additional dummy sub-tokens (i.e., dummy concentric circles) to each circular range query, and set the total number of sub-tokens as K to hide the actual number of concentric circles m, where $m \leq K$.

More specifically, for each circular range query, we can generate some dummy concentric circles, whose radiuses are beyond the data space, so that the points in the dataset will never be on the boundaries of dummy concentric circles. For example, if the data space for all the points in a real dataset is $\{x \in [0, 100]\} \wedge \{y \in [0, 100]\}$, we can generate a dummy concentric circle with $R = 200$. Similarly, radius privacy can also be conducted in CRSE-I by choosing a value of K to hide the actual value of m during the setup of CRSE-I.

The Challenge and Trade-off of Achieving Faster-Than-Linear Search. Since we do not use any data structures to index data, our schemes are linear search regarding to the number of data records. Some readers may argue our design is not efficient yet, especially on large-size datasets (e.g., millions or billions of data records). However, if we consider none of the previous searchable encryption schemes is able to particularly support circular range search on encrypted spatial data (*with a minimal one-round client-server interaction and a single cloud server*), our design is already a step forward. Besides, building a linear search scheme is normally a necessary stepping stone for future designs achieving faster-than-linear search, especially on encrypted data. For example, the previous searchable encryptions for common SQL queries, such as keyword search [4] and range search [13], [14], all take linear search as their first steps.

Generally, to obtain faster-than-linear search, a common approach (in the plaintext domain) is to leverage data structures. R-trees [17] are perhaps the most popular ones used for spatial data. In order to strictly follow the search algorithm of an R-tree (i.e., achieving faster-than-linear search) for circular range queries, besides testing whether a point is inside a circle on encrypted data at each leaf node in an R-tree (which we can support with our current design), we also need to verify *whether a rectangle intersects with a circle on encrypted data* at each non-leaf node, which unfortunately our design is not sufficient. Stated differently, besides checking whether a point is inside a circle, if we are able to find an efficient approach to verify whether a rectangle intersects with a circle in the ciphertext domain, then essentially we find a solution for building faster-than-linear circular range search.

On the other hand, since achieving faster-than-linear search needs to reveal additional meaningful information (i.e., whether a rectangle intersects with a circle), the security of a faster-than-linear search scheme will be **weaker** than what we have rigorously defined in our schemes. For instance, if a tree structure (e.g., an R-tree) will be used, the additional privacy leakage in a tree, such as *path pattern* [12], [17], *tree size&height*, *tree structures*, should also be properly considered in the security game and the leakage function. Moreover, *secure dynamic data with trees* has always been another major challenging issue. Meanwhile, the opportunity of using parallel computing in the cloud *may* be restricted as well due to the additional use of tree structures.

VII. Security Analysis

Security of CRSE-I. From the high-level, since CRSE-I can be interpreted as a specific instance of SSW with predefined form of vector $\vec{u}$ and $\vec{v}$, the security of CRSE-I under Selective Chosen-Plaintext Attacks can be proved based on the security of SSW [18] under Selective Chosen-Plaintext Attacks. Specifically, we have

Theorem 2: **(SCPA Query Privacy of CRSE-I)** *CRSE-I is query secure under Selective Chosen-Plaintext Attacks, if SSW is query secure under Selective Chosen-Plaintext Attacks.*

Proof: See the detailed proof in Appendix. ■

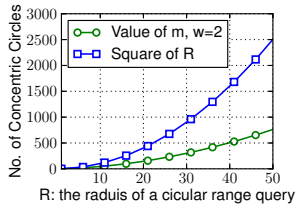

Fig. 9. Impact of R on the number of concentric circles.

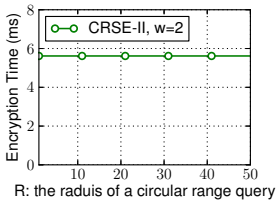

Fig. 10. Impact of R on encryption time (ms) per data record.

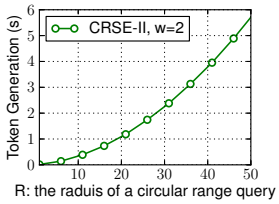

Fig. 11. Impact of R on token generation time (second) per query.

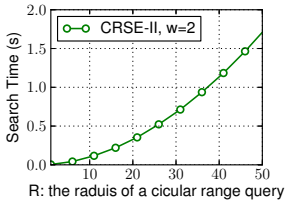

Fig. 12. Impact of R on search time (second) per data record.

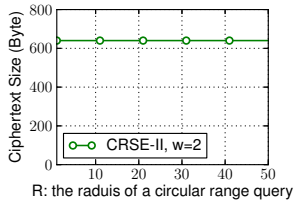

Fig. 13. Impact of R on the size of a ciphertext (Byte).

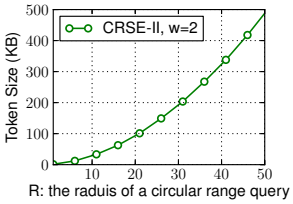

Fig. 14. Impact of R on search token size (KB) of a query.

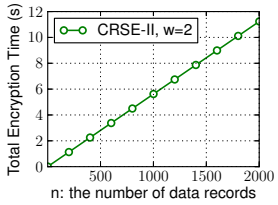

Fig. 15. Impact of n on total encryption time (second) on multiple data records.

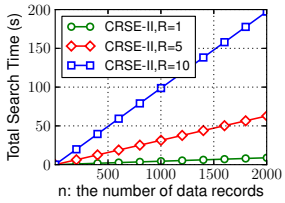

Fig. 16. Impact of n on total average search time (second) per query on multiple data records.

***Theorem** 3:* **(SCPA Data Privacy of CRSE-I)** *CRSE-I is data secure under Selective Chosen-Plaintext Attacks, if SSW is data secure under Selective Chosen-Plaintext Attacks.*

Proof: See further analyses in Appendix. ■

Security of CRSE-II. Although our second scheme is more efficient, and a permutation is utilized to enhance the privacy, its security is still weaker than what we rigorously defined for the first scheme. Essentially, the security weakness is introduced by the existence of sub-tokens in CRSE-II, where each sub-token can be independently used to evaluate a *match* on whether a point is on the boundary of a concentric circle.

In Appendix, we use a concrete example to describe the security weakness of CRSE-II. In addition, we show that CRSE-II can still be defined secure under Selective Chosen-Plaintext Attacks, but some additional restrictions (compared to Def. 2 and Def. 3) are needed to rigorously capture the corresponding security weakness. See details in Appendix.

VIII. Performance

In this section, we evaluate the performance of our proposed schemes in real cloud platform. We leverage the GNU Multiple Precision Arithmetic (GMP) and Pairing-Based Cryptography (PBC) library to test the running time of cryptographic operations in our schemes, and run our test code in an Amazon EC2 medium instance of Ubuntu 14.04 with variable ECUs (i.e., EC2 Compute Unit), 2 CPUs (2.5 GHz Intel Xeon Family) and 4 GB Memory. In addition, we choose super-singular curve $y^2 = x^3 + x$ to achieve the fastest performance in PBC for evaluating pairing operations, which are the dominating operations in our search process on encrypted spatial data. Specifically, the average running time of a paring operation with the preprocessing model in PBC is around 0.44 milliseconds in our experiments. Due to space limitations, we focus our results where the number of dimensions is $w = 2$.

Performance of CRSE-I. The complexity of CRSE-I is $O(\alpha^m)$ in the aspect of encryption time, token generation time and search time, where $\alpha = (w + 2)$ is the length of vectors, and the number of concentric circles m increases with $O(R^2)$.

R	m	Enc	GenToken	Search
1	2	0.015	0.019	0.009
2	4	0.077	0.102	0.050
3	7	3.09	4.12	1.96

TABLE I
RUNNING TIME (SECOND) OF CRSE-I WHEN $w = 2$.

R	m	Ciphertext Size	Search Token Size
1	2	2.18	2.18
2	4	32.90	32.90
3	7	2097.28	2097.28

TABLE II
CIPHERTEXT&SEARCH TOKEN SIZE (KB) OF CRSE-I WHEN $w = 2$.

Based on the sum of two square theorem, the actual value of m increasing with radius R under $w = 2$ is described in Fig. 9. In Table I and Table II, we present some results to show how CRSE-I dramatically increases with radius R in terms of running time and ciphertext&search token size.

Performance of CRSE-II. The performance of CRSE-II are described in Fig. 10 to Fig. 12. We can see that the encryption time of CRSE-II is independent with radius R. The token generation time and search time are both increasing with the square of radius R. Specifically, if $w = 2$ and $R = 10$, CRSE-II needs only 5.61 milliseconds to encrypt a data record, takes 329.47 milliseconds to generate a search token for a circular range query, and requires around 98.65 milliseconds (*in average case*) to check whether a point is inside the given circle on encrypted data. Similarly, as presented in Fig. 13 and Fig. 14, the size of a ciphertext in CRSE-II is independent with radius R, and the size of a search token increases with the square of radius R. More specifically, when $w = 2$ and $R = 10$, the size of a ciphertext is 640 Bytes, and the size of a search token is 28.16 KB. As we can see that, CRSE-II is much efficient compared to CRSE-I.

Performance of CRSE-II on Datasets. As we mentioned, it is obvious to see that the search time of CRSE-II on a dataset (i.e., multiple data records) are linearly increasing with the number of encrypted data records n. Specifically, in Fig. 16, if $R = 10$ and $w = 2$, the search time of CRSE-II with $n = 1,000$ is 98.65 seconds; while if $R = 1$ and $w = 2$,

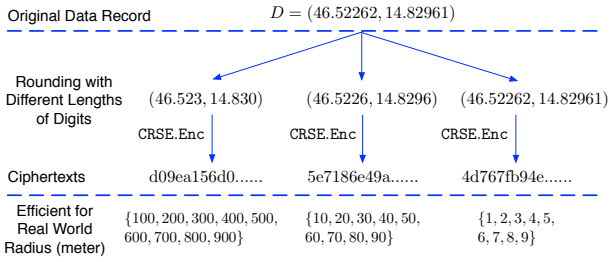

Fig. 17. An example of maintaining multiple ciphertexts for each data record with CRSE-II to support efficient search with different levels of data accuracy.

(Latitude, Longitude)	R	Real World Radius (meter)	Average Search Time (second)
(46.52262, 14.82961)	100	100	6165.50
(46.5226, 14.8296)	10	100	98.65
(46.523, 14.830)	1	100	4.44

TABLE III
TRADE-OFF BETWEEN EFFICIENCY AND DATA ACCURACY WHEN $w = 2$ AND $n = 1,000$ (THE REAL WORLD RADIUS IS AN APPROXIMATE RESULT).

the search time of CRSE-II on the same number of encrypted data records is only 4.44 seconds. As we can see, the total search time on a dataset is faster and more desirable when R is smaller.

Trade-off between Efficiency and Data Accuracy. One practical method to still guarantee an efficient search time for handling large circular range queries in real applications is to decrease data accuracy. We take a part of a dataset of users' location check-in information from an LBS (named Brightkite) as an example to show how this trade-off works, where the original dataset can be obtained from Stanford University's SNAP project [33].

For instance, given a point (i.e., a user's check-in location) $D = \{\texttt{Lat.} = 46.5226, \texttt{Long.} = 14.8296\}$ from the dataset, where its equivalent integer format is $\{465226, 148296\}$, we can search other points close to point D within a distance of approximate[3] 100 meters by setting $R = 10$ in CRSE-II. However, if we present the point as $D = \{\texttt{Lat.} = 46.523, \texttt{Long.} = 14.830\}$ (*rounding with one digit shorter*), where its equivalent integer format is $\{46523, 14830\}$, then we can still search points within approximate 100 meters by choosing $R = 1$, which performs circular range search within the same radius in the real world but significantly improves the search time on multiple data records (as presented in Table III). Said differently, a client is expected to spend more search time if it would like to receive more accurate search results.

In fact, we can maintain multiple ciphertexts for each data record with CRSE-II (an example is illustrated in Fig. 17) to flexibly support circular range queries with different levels of data accuracy and search efficiency. As a necessary trade-off, extra overheads are needed for computing and storing multiple ciphertexts of each data record. Moreover, since each encrypted data record in our schemes can be evaluated independently with a given search token, the performance of our design can be further improved by using *parallel computing* with multiple instances of Amazon EC2.

[3]The accurate distance based on location information can be computed using some online tool: http://www.movable-type.co.uk/scripts/latlong.html

IX. Conclusion and Future Work

We propose two novel schemes to support circular range search on encrypted spatial data without revealing data privacy or query privacy to the public cloud. Our future research include 1) designing circular range searchable encryption achieving faster-than-linear search with regard to the number of data records; 2) studying searchable encryption schemes for other common geometric queries, such as *simplex range search* (i.e., retrieving points that are inside a triangle).

References

[1] P. Agarwal and J. Erickson, "Geometric Range Searching and Its Relatives," *Discrete and Computational Geometry*, 1999.

[2] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, "Location Privacy via Private Proximity Testing," in *Proc. of NDSS'11*, 2011.

[3] Satyan L. Devadoss and Joseph O'Rourke, *Discrete and Computational Geometry*. Princeton University Press, 2011.

[4] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," in *Proc. of IEEE S&P'00*, 2000.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," in *Proc. of EUROCRYPT'04*, 2004.

[6] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic Searchable Symmetric Encryption," in *Proc. of ACM CCS'12*, 2012, pp. 965–976.

[7] J. Lai, X. Zhou, R. H. Deng, Y. Li, and K. Chen, "Expressive Search on Encrypted Data," in *Proc. of ACM ASIACCS'13*, 2013, pp. 243–251.

[8] 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. of ACM AISACCS'13*, 2013.

[9] E. Stefanov, C. Papamanthou, and E. Shi, "Practical Dynamic Searchable Encryption with Small Leakage," in *Proc. of NDSS'14*, 2014.

[10] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation," in *Proc. of NDSS'14*, 2014.

[11] W. K. Wong, B. Kao, D. W. Cheung, R. Li, and S. M. Yiu, "Secure Query Processing with Data Interoperability in a Cloud Database Environment," in *Proc. of ACM SIGMOD'14*, 2014.

[12] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovin, "Blind Seer: A Searchable Private DBMS," in *Proc. of IEEE S&P'14*, 2014.

[13] D. Boneh and B. Waters, "Conjunctive, Subset, and Range Queries on Encrypted Data," in *the Proceedings of Theory of Cryptography (TCC)*. Springer-Verlag, 2007, pp. 535–554.

[14] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig, "Multi-Dimensional Range Query over Encrypted Data," in *Proc. of IEEE S&P'07*, 2007, pp. 350–364.

[15] Y. Lu, "Privacy-Preserving Logarithmic-time Search on Encrypted Data in Cloud," in *Proc. of NDSS'12*, 2012.

[16] P. Wang and C. V. Ravishankar, "Secure and Efficient Range Queries on Outsourced Databases Using R-trees," in *Proc. of IEEE ICDE'13*, 2013.

[17] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li, "Maple: Scalable Multi-Dimensional Range Search over Encrypted Cloud Data with Tree-based Index," in *Proc. of ACM ASIACCS'14*, 2014.

[18] E. Shen, E. Shi, and B. Waters, "Predicate Privacy in Encryption Systems," in *Proc. of TCC'09*, 2009, pp. 457–473.

[19] R. Cannetti, S. Halevi, and J. Katz, "A Forward-Secure Public-Key Encryption Scheme," in *Proc. of EUROCRYP'03*, 2003.

[20] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order Preserving Encryption for Numeric Data," in *Proc. of ACM SIGMOD'04*, 2004.

[21] R. A. Popa, F. H. Li, and N. Zeldovich, "An Ideal-Security Protocol for Order-Preserving Encoding," in *Proc. of IEEE S&P'13*, 2013.

[22] W. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure KNN Computation on Encrypted Databases," in *Proc. ACM SIGMOD'09*, 2009, pp. 139–152.

[23] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing Private Queries over Untrusted Data Cloud through Privacy Homomorphism," in *Proc. IEEE ICDE*, 2011, pp. 601–612.
[24] Y. Elmehdwi, B. Samanthula, and W. Jiang, "Secure k-Nearest Neighbor Query over Encrypted Data in Outsourced Environments," in *Proc. of IEEE ICDE'14*, 2014.
[25] G. Zhong, I. Goldberg, and U. Hengartner, "Louis, Lester and Pierre: Three Protocols for Location Privacy," in *Proc. of PET'07*, 2007.
[26] J. Sedenka and P. Gasti, "Privacy-Preserving Distance Computation and Proxitity Testing on Earth, Done Right," in *Proc. of ACM ASIACCS'14*, 2014, pp. 99–110.
[27] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.
[28] P. Paillier, "Public-key Cryptosystems Based on composite Degree Residuosity Classes," in *Proc. of EUROCRYP'99*, 1999, pp. 223–238.
[29] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. CRC Press, 2007.
[30] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path ORAM: An Extremely Simple Oblivious RAM Protocol," in *Proc. of ACM CCS'13*, 2013.
[31] J. Katz, A. Sahai, and B. Waters, "Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products," in *Proc. of EUROCRYPT'08*, 2008, pp. 146–162.
[32] K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*. Springer, 1998.
[33] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and Mobility: User Movement in Location-Based Social Networks," in *Proc. of ACM KDD'11*, 2011.

APPENDIX

***Definition** 2:* **(SCPA Query Privacy).** *Let* $\Pi = (\texttt{GenKey}, \texttt{Enc}, \texttt{GenToken}, \texttt{Search})$ *be a symmetric-key CRSE scheme over security parameter* λ *and data space* Δ_T^w*:*

***Init:** The adversary submits two circular range queries* Q_0 *and* Q_1 *with the same radius to the challenger, where* $Q_0 = \{(x_0, y_0), R\}$*,* $Q_1 = \{(x_1, y_1), R\}$*,* $Q_0, Q_1 \subseteq \Delta_T^w$*.*

***Setup:** The challenger runs* $\texttt{GenKey}(1^\lambda, \Delta_T^w)$ *to generate a secret key* SK *and it keeps* SK *private.*

***Phase 1:** The adversary adaptively issues a number of requests, where each request is one of the two following types:*

- *Ciphertext Request: On the* j*th ciphertext request, the adversary outputs a data record* D_j*, where* $D_j \in \Delta_T^w$*. The challenger responses with a ciphertext* $C_j = \texttt{Enc}(\texttt{SK}, D_j)$*, where* D_j *is subjected to*
  1) $\mathcal{L}(D_j, Q_0) = \mathcal{L}(D_j, Q_1)$*;*
  2) $(D_j \in Q_0) \wedge (D_j \in Q_1)$ OR $(D_j \notin Q_0) \wedge (D_j \notin Q_1)$*;*

  with 1) AND 2)*.*
- *Token Request: On the* j*th token request, the adversary outputs a circular range query* $Q'_j = \{(x'_j, y'_j), R'_j\}$*, where* $Q'_j \subseteq \Delta_T^w$*. The challenger responses with a search token* $\texttt{TK}'_j = \texttt{GenToken}(\texttt{SK}, Q'_j)$*.*

Challenge:** With* Q_0*,* Q_1 *chosen in* ***Init*, the challenger flips a coin* $b \in \{0, 1\}$ *and returns* $\texttt{TK}_b = \texttt{GenToken}(\texttt{SK}, Q_b)$ *to the adversary.*

Phase 2:** The adversary continues to adaptively issue a number of requests, which are still subjected to the same restrictions in* ***Phase 1*.*

***Guess:** The adversary takes a guess* b' *of* b*.*

We say that scheme Π is SCPA query secure if for any polynomial time adversaries in the above game, it has at most negligible advantage

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\text{SCPA-Query}}(1^\lambda, \Delta_T^w) = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \texttt{negl}(\lambda)$$

where $\texttt{negl}(\lambda)$ denotes a negligible function in λ.

***Definition** 3:* **(SCPA Data Privacy).** *Let* $\Pi = (\texttt{GenKey}, \texttt{Enc}, \texttt{GenToken}, \texttt{Search})$ *be a symmetric-key CRSE scheme over security parameter* λ *and data space* Δ_T^w*:*

***Init:** The adversary submits two data records* D_0 *and* D_1 *with the same length to the challenger, where* D_0*,* $D_1 \in \Delta_T^w$*.*

***Setup:** The challenger runs* $\texttt{GenKey}(1^\lambda, \Delta_T^w)$ *to generate a secret key* SK *and it keeps* SK *private.*

***Phase 1:** The adversary adaptively issues a number of requests, where each request is one of the two following types:*

- *Ciphertext Request: On the* j*th ciphertext request, the adversary outputs a data record* D'_j*, where* $D'_j \in \Delta_T^w$*. The challenger responses with a ciphertext* $C'_j = \texttt{Enc}(\texttt{SK}, D'_j)$*.*
- *Token Request: On the* j*th token request, the adversary outputs a circular range query* $Q_j = \{(x_j, y_j), R_j\}$*, where* $Q_j \subseteq \Delta_T^w$*. The challenger responses with a search token* $\texttt{TK}_j = \texttt{GenToken}(\texttt{SK}, Q_j)$*, where* Q_j *is subjected to*
  1) $\mathcal{L}(D_0, Q_j) = \mathcal{L}(D_1, Q_j)$*;*
  2) $(D_0 \in Q_j) \wedge (D_1 \in Q_j)$ OR $(D_0 \notin Q_j) \wedge (D_1 \notin Q_j)$*;*

  with 1) AND 2)*.*

Challenge:** With* D_0*,* D_1 *chosen in* ***Init*, the challenger flips a coin* $b \in \{0, 1\}$ *and returns* $C_b = \texttt{Enc}(\texttt{SK}, D_b)$ *to the adversary.*

Phase 2:** The adversary continues to adaptively issue a number of requests, which are still subjected to the same restrictions in* ***Phase 1*.*

***Guess:** The adversary takes a guess* b' *of* b*.*

We say that scheme Π is SCPA data secure if for any polynomial time adversaries in the above game, it has at most negligible advantage

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\text{SCPA-Data}}(1^\lambda, \Delta_T^w) = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \texttt{negl}(\lambda)$$

where $\texttt{negl}(\lambda)$ denotes a negligible function in λ.

A. Proof of SCPA Query Privacy of CRSE-I

The main idea of the following proof is to simulate the query security game under SCPA defined in Def. 2 with an adversary $\mathcal{A}$ in the query security game of SSW under SCPA. And we prove that if $\mathcal{A}$ is able to distinguish two queries Q_0 and Q_1 in CRSE-I, it is equivalent for $\mathcal{A}$ to distinguish two corresponding vectors $\vec{v}_0$ and $\vec{v}_1$ in SSW, which contradicts to the assumption that SSW is query secure under SCPA. Specifically, we have

Init: The adversary $\mathcal{A}$ selects two circular range queries Q_0, Q_1 with the same radius R, where Q_0, $Q_1 \subseteq \Delta_T^w$, computes $\{m, r_1, ..., r_m\} \leftarrow \texttt{GenConCircle}(R)$, calculates polynomial

$P = P_1 \cdots P_m$, computes $\{\alpha, f_{\vec{u}}, f_{\vec{v}}\} \leftarrow \texttt{Split}(P)$, transforms $\vec{v}_0 = f_{\vec{v}}(Q_0)$, $\vec{v}_1 = f_{\vec{v}}(Q_1)$, and submits $\vec{v}_0$, $\vec{v}_1 \in \Delta_T^\alpha$ to the challenger.

Setup: The challenger runs $\texttt{SSW.GenKey}(1^\lambda, \Delta_T^\alpha)$ to generate a secret key $\texttt{SK}$ and keeps it private.

Phase 1: The adversary adaptively requests a number of queries. For *Ciphertext Query*, $\mathcal{A}$ outputs a data record D_j, where $D_j \in \Delta_T^w$, submits $\vec{u}_j = f_{\vec{u}}(D_j) \in \Delta_T^\alpha$ to the challenger. The challenger responses with a ciphertext $C = \texttt{SSW.Enc}(\texttt{SK}, \vec{u}_j)$ subjected with the following restrictions:

$$\begin{cases} \text{Either } (\vec{u}_j \circ \vec{v}_0 = 0) \wedge (\vec{u}_j \circ \vec{v}_1 = 0) \\ \text{or } (\vec{u}_j \circ \vec{v}_0 \neq 0) \wedge (\vec{u}_j \circ \vec{v}_1 \neq 0), \end{cases}$$

which are *equivalently* transformed from the original restrictions defined in Def. III:

$$\begin{cases} 1) \quad \mathcal{L}(D_j, Q_0) = \mathcal{L}(D_j, Q_1); \\ 2) \quad \text{Either } (D_j \in Q_0) \wedge (D_j \in Q_1) \text{ or } (D_j \notin Q_0) \wedge (D_j \notin Q_1). \end{cases}$$

Note that the leakage of radius pattern in leakage function $\mathcal{L}$ of CRSE-I maps to the *inevitable* leakage of the length of vector $\vec{v}_0$ and $\vec{v}_1$ in the selective security game of SSW [18].

For *Token Query*, $\mathcal{A}$ outputs a circular range query Q'_j, where $Q'_j \subseteq \Delta_T^w$, submits $\vec{v}'_j = f_{\vec{v}}(Q'_j) \in \Delta_T^\alpha$. The challenger responses with a search token $\texttt{TK}'_j = \texttt{SSW.GenToken}(\texttt{SK}, \vec{v}'_j)$.

Challenge: With $\vec{v}_0$, $\vec{v}_1$ selected in **Init**, the challenger flips a coin $b \in \{0, 1\}$, and returns $\texttt{TK}_b = \texttt{SSW.GenToken}(\texttt{SK}, \vec{v}_b)$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ continues to request a number of queries by following the same restrictions in **Phase 1**.

Guess: $\mathcal{A}$ takes a guess b' of b.

Since we successfully simulate query security game of CRSE-I under SCPA with an adversary $\mathcal{A}$ in the query security game of SSW under SCPA, if adversary $\mathcal{A}$ is able to distinguish Q_0 and Q_1 in CRSE-I, it is equivalent for it to distinguish $\vec{v}_0$ and $\vec{v}_1$ in SSW, which can be interpreted as:

$$\mathbf{Adv}^{\text{SCPA-Query}}_{\text{CRSE-I},\mathcal{A}}(1^\lambda, \Delta_T^w) = \mathbf{Adv}^{\text{SCPA-Query}}_{\text{SSW},\mathcal{A}}(1^\lambda, \Delta_T^\alpha) \leq \texttt{negl}(\lambda)$$

Therefore, CRSE-I is query secure under SCPA.

B. Proof of SCPA Data Privacy of CRSE-I

Similarly as the proof for SCPA query privacy, the main idea of this proof is to simulate the data security game under SCPA defined in Def. 3 with an adversary $\mathcal{A}$ in the data security game of SSW under SCPA. If adversary $\mathcal{A}$ is able to distinguish D_0 and D_1 in CRSE-I, it is able to distinguish $\vec{u}_0$ and $\vec{u}_1$ in SSW, where $\vec{u}_0 = f_{\vec{u}}(D_0)$, $\vec{u}_1 = f_{\vec{u}}(D_1)$, which indicates:

$$\mathbf{Adv}^{\text{SCPA-Data}}_{\text{CRSE-I},\mathcal{A}}(1^\lambda, \Delta_T^w) = \mathbf{Adv}^{\text{SCPA-Data}}_{\text{SSW},\mathcal{A}}(1^\lambda, \Delta_T^\alpha) \leq \texttt{negl}(\lambda)$$

We skip the details due to the limitation of space.

C. Security Analysis of CRSE-II

In the following, we first use an example to show the security weakness of CRSE-II, and capture this weakness consistently and rigorously in the security games under SCPA but with additional restrictions.

More specifically, assume only ciphertext C_0, C_1 and a permuted search token $\mathbf{TK}^* = (\texttt{TK}^*_1, ..., \texttt{TK}^*_m)$ are given. In addition, assume C_0 and C_1 are both introduced a match by $\mathbf{TK}^*$, but on two different sub-tokens. Then, these two ciphertexts of the two data records (D_0 and D_1) are *so far* computationally indistinguishable in CRSE-II. It is because both of the two cases shown in Fig. 18 could happen due to the additional use of permutation on all the m sub-tokens.

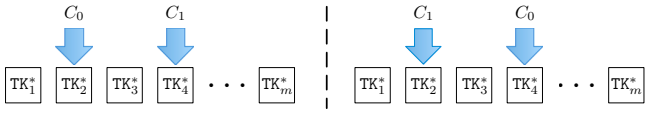

Fig. 18. CRSE-II: An Example with **Indistinguishability** between C_0, C_1.

However, if the adversary $\mathcal{A}$ still adaptively issues a ciphertext request based on a data record D'_j by following the original restrictions defined in Def. 3, where assuming C'_j (the ciphertext of D'_j) introduces a match on the same sub-token as C_0. According to the design of CRSE-II, this match not only indicates D'_j and D_0 are both inside the same circle, but also further reveals D'_j and D_0 are on the boundary of the same concentric circle. Since D'_j, D_0 and D_1 are chosen by the adversary in the security game, it immediately distinguishes C_0 and C_1 with the help of C'_j (as shown in Fig. 19), even though the permutation is used on all the m sub-tokens.

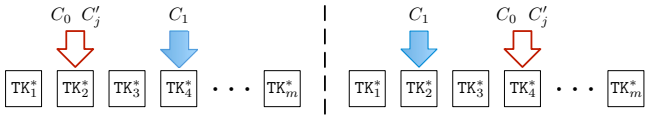

Fig. 19. An Example of CRSE-II with **distinguishability** between C_0, C_1.

Therefore, CRSE-II is not data secure under SCPA with the similar cases above. In fact, we can capture this weakness consistently in the security games under SCPA by adding additional restrictions in the **Phase 1** compared to the original definitions in Def. 3 and Def. 2. Specifically, with the two submitted points D_0 and D_1 in **Init**, the **Phase 1** of the data security game of CRSE-II under SCPA are described as:

Phase 1: *The adversary adaptively issues a number of requests, where each request is one of the two following types:*

- *Ciphertext Request: On the jth ciphertext request, the adversary outputs a data record D'_j, where $D'_j \in \Delta_T^w$. The challenger responses with a ciphertext $C'_j = \texttt{Enc}(\texttt{SK}, D'_j)$ is subjected to the following restriction:*
  1) *If $(D_0 \in Q_i) \wedge (D_1 \in Q_i)$, then $D'_j \notin Q_i$, for any previous token request of Q_i,*
- *Token Request: On the jth token request, the adversary outputs a circular range query Q_j, where $Q_j \subseteq \Delta_T^w$. The challenger responses with a search token $\texttt{TK}_j = \texttt{GenToken}(\texttt{SK}, Q_j)$, where Q_j is subjected to the following restrictions:*
  1) $\mathcal{L}(D_0, Q_j) = \mathcal{L}(D_1, Q_j)$;
  2) $(D_0 \in Q_j) \wedge (D_1 \in Q_j) \wedge (D'_i \notin Q_j)$ $\texttt{OR}$ $(D_0 \notin Q_j) \wedge (D_1 \notin Q_j)$, *for any previous ciphertext request of D'_i;*

  with 1) $\texttt{AND}$ 2).

Except **Phase 1**, the rest descriptions of this data security game under SCPA are as the same as the ones in Def. 3.

Similarly, we can also capture the corresponding weakness for query privacy by describing the query security game of CRSE-II under SCPA with additional restrictions in **Phase 1** based on Def. 2. Specifically, with the two submitted queries Q_0 and Q_1 in **Init**, we have

Phase 1: *The adversary adaptively issues a number of requests, where each request is one of the two following types:*

- *Ciphertext Request: On the jth ciphertext request, the adversary outputs a data record D_j, where $D_j \in \Delta_T^w$. The challenger responses with a ciphertext $C_j = \texttt{Enc}(\texttt{SK}, D_j)$ is subjected to the following restriction:*
  1) $\mathcal{L}(D_j, Q_0) = \mathcal{L}(D_j, Q_1)$;
  2) $(D_j \in Q_0) \wedge (D_j \in Q_1) \wedge (D_i' \notin Q_0) \wedge (D_i' \notin Q_1)$ `OR` $(D_j \notin Q_0) \wedge (D_j \notin Q_1)$, *for any previous ciphertext request of D_i';*

  with 1) `AND` 2).
- *Token Request: On the jth token request, the adversary outputs a circular range query Q_j', where $Q_j' \subseteq \Delta_T^w$. The challenger responses with a search token $\texttt{TK}_j = \texttt{GenToken}(\texttt{SK}, Q_j')$.*

Except **Phase 1**, the rest descriptions of this query security game under SCPA are as the same as the ones in Def. 2.

Since CRSE-II is essentially built based on SSW as well, we can still use similar approaches as we did in the security proofs of CRSE-I to demonstrate the security of CRSE-II. Specifically, we can still simulate the security games of CRSE-II with an adversary $\mathcal{A}$ in the security games of SSW, and prove that if adversary $\mathcal{A}$ is able to break the security games, it is equivalent to break the security games of SSW, which contradicts to the assumption that SSW is secure in terms of data privacy and query privacy. Due to space limitations, we skip further details.

D. Security Analysis on a Dataset

The encryption with our design on a dataset (i.e., multiple data records) is essentially a case of **multiple encryptions**. And the security of the multiple encryptions of CRSE-I or CRSE-II can be proved based on the following claim from the textbook (Chapter 3, Claim 3.23) [29].

***Claim** 1: Any symmetric-key encryption scheme that has indistinguishable encryptions under a Chosen-Plaintext Attack also has indistinguishable multiple encryptions under a Chosen-Plaintext Attack.*

Specifically, since CRSE-I (or CRSE-II) is data secure and query secure under Selective Chosen-Plaintext Attacks (which is a weak version of Chosen-Plaintext Attacks) and it is symmetric-key based, then the multiple encryptions of CRSE-I (or CRSE-II) on a dataset is also data secure and query secure under Selective Chosen-Plaintext Attacks.