Searchable Encryption for Geo-Referenced Data

Oriol Farràs
Universitat Rovira i Virgili
Tarragona, Catalonia, Spain
Email: oriol.farras@urv.cat

Jordi Ribes-González
Universitat Rovira i Virgili
Tarragona, Catalonia, Spain
Email: jordi.ribes@urv.cat

***Abstract*—Searchable encryption schemes allow to query on encrypted data in a secure way. In this work we present different techniques that achieve range queries in searchable encryption, thus enhancing performance and reducing the leakage of information. We focus on the case of one and two-dimensional range queries to an encrypted geo-referenced database in a client-server architecture.**

***Index Terms*—Cryptography, Searchable Encryption, Functional Encryption, Geo-Referenced Data, Cloud Computing.**

I. Introduction

The cloud computing paradigm offers data storage and data computation services in external servers that lead to many economic benefits. However, many users are reluctant to outsource data storage and data computation to the cloud because of security and privacy concerns. Regarding these issues, there are still many challenges to be solved, but in the recent years there have been important advances in cryptographic techniques that allow to take advantage of the cloud benefits while securing the data.

The main problem we deal with in this work is searching on encrypted data. In particular, we consider the case in which we want to perform queries on encrypted geo-referenced data. This is motivated by the natural issue of users owning datasets containing geo-referenced data. For example, data about geologic resources, water supplies, critical infrastructures or civil constructions. Typically, this information is stored and indexed locally. But lately, due to economical reasons, users want to outsource the data to the cloud. Since the information is valuable and confidential, they want to prevent the server and external attackers from obtaining information about it.

A simple approach for secure storage is to encrypt all the data in a straightforward way, using symmetric key encryption schemes. However, this approach provides very limited functionalities because general encryption techniques do not allow searching or computing on the encrypted data. Hence, we consider a solution based on *searchable encryption*, a cryptographic primitive that allows to remotely query on outsourced encrypted data in a secure way. Song et al. [11] presented one of the first searchable encryption schemes in 2000. Since their pioneering work, the field has expanded in many directions, by enhancing the security and efficiency of the schemes, by improving expressiveness of the queries, or by dealing with multi-user settings.

978-1-5090-1983-0/16/$31.00

In the client-server setting we are interested in, the client first creates an index of the data according to certain keywords and stores it along with the data in an encrypted form. Later, if the client wants to recover the elements of the database that contain some keywords, it sends a token associated to these keywords to the server, and the server returns the encrypted documents containing the keywords. We assume that the server behaves in an honest-but-curious manner, that is, the server always runs the protocol correctly, but it collects additional information and it is able to perform computations using the data it already holds.

In this work we provide techniques for searchable encryption that reduce the communication and computational costs of range queries, and techniques that reduce the leakage of information. This work is focused on two-dimensional range geo-referenced queries.

Due to restrictions of the setting, we study symmetric-key cryptographic schemes. We provide general techniques that can be used for a broad family of schemes, the ones supporting at least single keyword searches. Moreover, we also provide techniques that take advantage of the properties of more advanced schemes supporting Boolean queries, like the one of Cash et al. [4]. Recently, Faber et al. [6] presented an extension of the scheme in [4] that supports range queries. Following [6] we provide an adaptation of their scheme to two-dimensional range queries. In the public-key setting, there exist schemes supporting range queries [1], [10], but these schemes do not fit our setting because we just handle the one writer and one reader scenario.

Order preserving encryption schemes (as those used in CryptDB [9], for example) provide a very efficient solution for one-dimensional range queries, but the security notion such these schemes is weaker than the standard security notions of searchable encryption. Recent studies show that the leakage can be very significant [8].

In Section II we give an introduction to searchable encryption, providing a general description of the functions involved. In Section III we suggest different representations of geo-referenced data, and we provide the associated SE constructions in Section IV. The efficiency and the information leakages associated to these options are described in Sections V and VI, and the conclusions and future work are sketched in Section VII.

II. Problem Definition

A. Geo-Spatial Data

We study the problem of searching on encrypted data in the context of the project CLARUS: *A framework for user centered privacy and security in the cloud* [5]. The main objective of CLARUS is to develop a secure framework for storing and processing data outsourced to the cloud, so end-users can monitor, audit and control their stored data while gaining the cost-saving benefits that cloud services bring. A key case study in the project is about outsourcing confidential data containing the location of critical natural earth resources as water boreholes, minerals, or rare earth materials.

We focus on the two-dimensional range queries that ask for the collection of features in the database intersecting a given rectangle. The problem is rather general, and our approach can be applied to multi-dimensional range queries and it is independent of the structure of the dataset.

We consider the requirements derived from the use of the OGC® Web Feature Service Interface Standard (WFS) interface. WFS defines web interface operations for querying and editing vector geographic features. In this case the user requests geographical features across the web using platform-independent calls. These features can be described by points, paths or polygons. Our queries can be described using PostGIS / PostgreSQL. PostGIS is a spatial database extender that adds support for geographic objects allowing location queries. In this work, we enhance the security of `SELECT` queries with clause `WITH ST_MakeEnvelope(X,Y,*)`, which creates a rectangle described by the two opposite vertexes `X` and `Y` and returns the elements intersecting this rectangle.

B. Searchable Encryption

The geo-referenced data-oriented solutions presented in this article make use of an underlying searchable symmetric encryption scheme. Such schemes allow a client to outsource an encrypted version of a dataset to a semi-trusted, honest-but-curious server, while preserving basic searching capabilities over the encrypted data.

We consider the one writer and one reader setting. Firstly, a single client wants to upload a dataset $\mathbf{D}$ to a server. The dataset is composed of tuples (W_i, D_i), where the *document* D_i is any kind of plaintext (text, audio, images, etc.) and W_i is a list of *keywords* associated D_i.

A SSE scheme $\mathcal{S}$ first provides a client with a symmetric key k by executing an algorithm $\mathcal{S}.\mathrm{Gen}$. Using this key k, an algorithm $\mathcal{S}.\mathrm{Enc}$ allows the client to build an encryption $\mathbf{C}$ of the dataset $\mathbf{D}$. Additionally, this algorithm creates a *searchable index* I in the process, which is uploaded to the server along with the ciphertexts $\mathbf{C}$.

Afterwards, the client wants to retrieve the documents satisfying a particular predicate f on keywords. A predicate is specified by a Boolean formula on a tuple of keywords. For example, the predicate $w_1 \wedge (w_2 \vee w_3)$ is satisfied by documents whose associated keyword list contains the keyword w_1 and either w_2 or w_3.

The client furnishes a predicate f to the algorithm $\mathcal{S}.\mathrm{Trpdr}$ to generate a *trapdoor* (or *search token*) T encoding the given predicate. To query for documents satisfying predicate f, the client sends T to the server.

The server then combines the trapdoor T and the searchable index I using the public algorithm $\mathcal{S}.\mathrm{Search}$. In this way it obtains a tuple of ciphertexts, which are the encrypted documents satisfying the predicate that T represents.

We present below a general model for searchable symmetric encryption schemes. For an introduction to searchable encryption, see [2].

Definition II.1. A searchable symmetric encryption (SSE) scheme $\mathcal{S}$ consists of four polynomial-time algorithms:

$\mathcal{S}.\mathrm{Gen}(\lambda)$:
: Probabilistic algorithm run by the client that, given a security parameter λ, returns a secret key k and the public parameters params of the scheme, including a symmetric encryption scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$.

$\mathcal{S}.\mathrm{Enc}_k(\mathbf{D})$:
: Probabilistic algorithm run by the client and taking as input a tuple of plaintexts $\mathbf{D} = (W_i, D_i)_{i=1}^N$, where W_i is a tuple of keywords attached to the document D_i. It returns an encrypted index I together with a tuple of encrypted documents $\mathbf{C} = (C_i)_{i=1}^N = (\mathrm{Enc}_k(D_i))_{i=1}^N$ for the algorithm Enc in params.

$\mathcal{S}.\mathrm{Trpdr}_k(f)$:
: Algorithm run by the client that takes a predicate on keywords f as input and returns a trapdoor (search token) T associated to f.

$\mathcal{S}.\mathrm{Search}(I, \mathbf{C}, T)$:
: Deterministic algorithm run by the server and taking as input an encrypted index I, a tuple of ciphertexts $\mathbf{C}$ and a trapdoor T. It returns a tuple of elements of $\mathbf{C}$.

$\mathcal{S}.\mathrm{Dec}_k(C)$:
: Deterministic algorithm run by the client and taking an encrypted document C as input. It returns the output of $\mathrm{Dec}_k(C)$ for the algorithm Dec in params.

If the algorithm $\mathcal{S}.\mathrm{Trpdr}$ supports only single keywords as predicates, we say that $\mathcal{S}$ is a *single-keyword* SSE scheme. Alternatively, if $\mathcal{S}.\mathrm{Trpdr}$ supports arbitrary conjunctions of such predicates, we say that $\mathcal{S}$ is a *conjunctive* SSE scheme. Finally, if $\mathcal{S}.\mathrm{Trpdr}$ supports arbitrary Boolean formulas, we say that $\mathcal{S}$ is a *Boolean* SSE scheme.

From all searchable symmetric encryption schemes fitting this definition, we choose the OXT scheme by Cash et al. [4] as the underlying scheme for the instantiations of our proposed schemes. The OXT scheme is suitable for static databases and it provides highly efficient Boolean search while incurring moderate and quantifiable leakage to the server. However, we provide general techniques that can also be used with any searchable encryption scheme, even if it does not support conjunctive or Boolean queries.

Regarding security, queries using searchable encryption induce some leakage on the information that is exchanged

or stored. For example, if the function $\mathcal{S}.\mathrm{Search}$ returns the same ciphertexts for two different queries, it reveals that the two queries are equivalent in $\mathbf{D}$.

Several works define different models of leakage, depending on the scheme and the followed adversary model. The notion of security of searchable symmetric encryption schemes is conditioned to a predefined leakage function. In [4], Cash et al. define the concept of $\mathcal{L}$-semantic security against adaptive attacks, and in this work we base our security analysis on their security definition. Further, in [4, Section 5.3], they show precise upper bounds on the allowed leakage of the OXT scheme. This enables us to provide a clean security analysis in Section V by describing upper bounds on the leakage incurred by each instantiation of our proposed schemes.

III. Searchable Encryption for Geo-Referenced Data

In this section we present the structure for geo-referenced data considered in this work. We use binary tree and quadtree structures in order to describe the geometrical objects used in the proposed schemes. Next, we define 2-dimensional range searchable symmetric encryption schemes.

A. Data Model

In our model, the client owns a collection of documents, each of which is attached to a particular geographical location in some predefined area. Without loss of generality, we assume that the predefined area where the searches are performed is the two-dimensional grid $\Lambda_n = [0, 2^{n+1} - 1] \times [0, 2^{n+1} - 1]$ for some fixed bit length $n + 1 > 0$.

The tuples $(a, b) \in \Lambda_n$ are called *points*. By an abuse of notation, we also denote points in Λ_n by $(a_n \cdots a_0, b_n \cdots b_0)$, where $a_n \cdots a_0$ and $b_n \cdots b_0$ are the *bit representations* of a and b. For instance, we can consider the point $(1001, 0100) \in \Lambda_3$.

We restrict our model to the case where each document is attached to a single point in Λ_n. Since all shapes in Λ_n can be represented as some set of points, the constructions naturally extend to cases where documents are attached to lines and polygons. We comment this extension in Section VII.

Let $\mathcal{I}_n = \{[a, b] : 0 \leq a \leq b < 2^{n+1}, a, b \in \mathbb{Z}\}$ denote the set of *intervals* (or *ranges*) with only $n+1$-bit values. We call the elements in $\mathcal{I}_n^2 = \mathcal{I}_n \times \mathcal{I}_n$ *rectangles*. For instance, the rectangle $[0100, 1011] \times [1000, 1111] \in \mathcal{I}_3^2$ is a product of two intervals of length 8.

In our setting, the client first delegates an encrypted version of its dataset to a server. Afterwards, it wants to issue a query to the server and retrieve the subset of the outsourced dataset whose associated points fall inside a chosen location. Throughout the article we restrict such queried locations to be rectangles, that is, the client only asks for the set of documents whose points lie in rectangles of its choice.

B. Prefix Decomposition

1) Binary Trees and Prefixes: Binary trees have been widely applied in computer science, with applications ranging from routing protocols to sorting and text searching. In the context of searchable encryption, they are used to achieve security in range queries over encrypted data in works such as [10] and, recently, in [6].

Let $0 \leq a < 2^{n+1}$ an integer. Then a can be represented with $n + 1$ bits as $a_n \cdots a_0$. It is clear that a can be viewed as a leaf of the full binary tree of height $n + 1$, in such a way that the parent of a at depth i is associated to the i most significant bits of a. In this way, we denote the parent of a at depth $i \leq n$ with the binary representation $a_n \cdots a_{n-i+1}*$, where the wildcard $*$ denotes an arbitrary suffix. We call such representation a *prefix* of *length* i.

The *parent tuple* of a non-negative integer a with representation $a_n \cdots a_0$ is the tuple of prefixes defined by the parents of a in increasing depth, that is, $(a_n*, a_n a_{n-1}*, \ldots, a_n a_{n-1} \cdots a_1*, a_n a_{n-1} \cdots a_1 a_0)$. For instance, the parent tuple of 0110 is $(0*, 01*, 011*, 0110)$. We use this tuple to generate the indexed element associated to a point, as in [10], [6].

We say that an interval is a *prefix range* if it is expressible as a prefix. For example, in a 4-bit field, the prefix $01*$ defines the prefix range $[0100, 0111]$. Clearly, any prefix of length i defines a prefix range of length 2^{n-i+1}.

Given an arbitrary interval, consider an expression of it as a disjoint union of some set of prefix ranges. We call the set of corresponding prefixes a *prefix decomposition* of the interval. For example, the interval $[0011, 1101]$ admits a decomposition into 4 prefixes, namely $\{0011, 01*, 10*, 110*\}$.

Given an arbitrary interval, consider a set of prefix ranges whose union contains the interval. We call the set of corresponding prefixes an *over-cover* of the interval. For example, the interval $[0011, 1101]$ admits an over-cover $\{0011, 01*, 1*\}$. This notion was first defined in [6].

As observed in [10, Section 5.1], an element belongs to an interval if and only if the intersection between the parent tuple of the element and any prefix decomposition of the interval is nonempty. Similarly, if an element belongs to an interval, the intersection between the parent tuple of the element and any over-cover of the interval is nonempty. Thanks to this property, in Sections IV-A,IV-C we use prefix decompositions and over-covers in order to generate queries for ranges and rectangles.

2) Quadtrees and Prefixes: As we see in Section V or in works such as [10], [6], the usage of binary trees in 2-dimensional range searchable encryption induces some privacy problems. The main reason behind is that the plaintext documents use a binary tree for each coordinate, so the dimensions are decoupled. In an attempt to fix this, we develop the same concepts above by using *quadtrees* instead of binary trees.

A *quadtree* is a full 4-ary tree. Most common uses of quadtrees involve two-dimensional data, such as in image representation, spatial indexing or storing matrix information. Geometrically, they are usually interpreted as representing a recursive subdivision of space into quadrants.

Let (a, b) be a point in Λ_n with binary representation $(a_n \cdots a_0, b_n \cdots b_0)$. Then (a, b) can be viewed as a leaf of

a quadtree of height $n+1$, in such a way that the parent of (a,b) at depth i is associated to the i most significant bits of a and the i most significant bits of b. In this way, we denote the parent of (a,b) at depth $i \leq n$ with the binary representation $(a_n a_{n-1} \cdots a_{n-i+1}*, b_n b_{n-1} \cdots b_{n-i+1}*)$, where the wildcard $*$ denotes an arbitrary suffix. We call such representation a *prefix* of *length* $2i$.

The parent tuple of the point $(a_n \cdots a_0, b_n \cdots b_0)$ is defined just as above. For instance, the parent tuple of $(001, 100)$ is $((0*, 1*), (00*, 10*), (001, 100))$.

We say that a rectangle $[a,b] \times [c,d]$ is a *prefix rectangle* if it is expressible as a prefix. This happens exactly when both $[a,b]$ and $[c,d]$ are prefix ranges expressible as prefixes of the same length. For example, when using 6-bit fields, the prefix $(110*, 011*)$ defines the prefix rectangle $[110000, 110111] \times [011000, 011111]$.

Of course, any rectangle can be expressed as a disjoint union of some prefix rectangles. We call the set of corresponding prefixes a *prefix decomposition* of the rectangle. For example, the rectangle $[00, 01] \times [00, 10]$ admits a decomposition into 3 prefixes, namely $\{(0*, 0*), (00, 10), (01, 10)\}$. In Section IV-B we state an upper bound on the minimal size of such prefix decompositions.

Given an arbitrary rectangle, consider a set of prefix rectangles whose union contains the rectangle. We call the set of corresponding prefixes an *over-cover* of the rectangle. For example, the rectangle $[00, 01] \times [00, 10]$ admits an over-cover $\{(0*, 0*), (0*, 1*)\}$.

Note that the parent tuple of a point consists exactly of the prefixes whose prefix rectangle contains that point. This is why a point belongs to a rectangle if and only if the intersection between the parent tuple of the point and any prefix decomposition of the rectangle is nonempty. Similarly, if a point belongs to a rectangle, then the intersection between the parent tuple of the point and any over-cover of the rectangle is nonempty. Thanks to this property, in Sections IV-B,IV-D we use prefix decompositions and over-covers in order to generate the query associated to a rectangle.

C. Searchable Encryption for Geo-Referenced Data

We start by giving context for our geo-referenced data-oriented solutions. Our aim is to provide schemes that enable a client to delegate an encrypted version of a geo-referenced dataset to a semi-trusted, honest-but-curious server, in such a way that searching capabilities over the encrypted data are preserved.

Assume that a client wants to upload a dataset $\mathbf{D}$ to a server. The dataset is composed of tuples $((w_{i,1}, w_{i,2}), D_i)$, where the document D_i is any kind of plaintext (text, audio, images, etc.) and $(w_{i,1}, w_{i,2})$ is the point in Λ_n where D_i is located.

A 2-dimensional range searchable symmetric encryption scheme $\mathcal{R}$ first provides a client with a symmetric key k by executing an algorithm $\mathcal{R}.\mathrm{Gen}$. Using this key k, an algorithm $\mathcal{R}.\mathrm{Enc}$ allows the client to build an encryption $\mathbf{C}$ of the dataset $\mathbf{D}$. Additionally, this algorithm creates a *searchable index* I in the process, which is uploaded to the server along with the ciphertexts $\mathbf{C}$.

Afterwards, the client wants to retrieve the documents located in a rectangle $R \in \mathcal{I}_n^2$. The client furnishes this rectangle to the algorithm $\mathcal{R}.\mathrm{Trpdr}$ to generate a *trapdoor* (or *search token*) T that encodes the given rectangle. To query for rectangle R, the client sends T to the server.

The server then combines the trapdoor T and the searchable index I by using the public algorithm $\mathcal{R}.\mathrm{Search}$. In this way, it obtains a tuple of ciphertexts, which are encryptions of all documents located in rectangle R.

We now define 2-dimensional range searchable symmetric encryption scheme.

Definition III.1. A 2-dimensional range searchable symmetric encryption (2-DRSSE) scheme $\mathcal{R}$ is a SSE scheme where

1) The output of $\mathcal{R}.\mathrm{Gen}(\lambda)$ includes a bit-length $n+1$.
2) The input of $\mathcal{R}.\mathrm{Enc}_k$ is a collection $\mathbf{D}$ of plaintext documents, where each document consists of a point in Λ_n and some attached plaintext (e.g., the id of some document), that is, $\mathbf{D} = ((w_{i,1}, w_{i,2}), D_i)_i$.
3) The input of $\mathcal{R}.\mathrm{Trpdr}_k$ is a rectangle $R \in \mathcal{I}_n^2$, seen as the disjunction of all points in R.

In this article, we use a SSE scheme $\mathcal{S}$ as a primitive for building 2-DRSSE schemes.

IV. Two-Dimensional Range Queries

In this section we describe our constructions. We start by presenting a solution based on prefix decompositions by using binary trees, and then we introduce a similar solution based on prefix decompositions by using quadtrees. Next, we admit a higher false-positive rate by dealing with over-covers, first by using binary trees and finally by using quadtrees.

Throughout the section, fix $n+1$ a positive bit length, so that all documents are located at points of Λ_n.

A. Prefix Decomposition Using Binary Trees

In [7] it is proved that, for $n \geq 2$, any interval in $\mathcal{I}_n$ admits a prefix decomposition using at most $2n$ prefixes. We next present an extension to this result, and we provide a proof in the full version of this article.

Theorem IV.1. *Let t be a positive integer. Then, any interval $[a,b]$ of length at most 2^t decomposes into at most $\max(t+1, 2t-1)$ prefixes.*

A procedure preDec to compute the prefix decomposition of an interval $[a,b] \in \mathcal{I}_n$ follows directly from the constructive argument in the proof of Theorem IV.1. We describe the algorithm in the full version of this article.

When encrypting a document located at a point (w_1, w_2), the idea behind the first scheme we propose is to attach the document to every possible tuple (w_1', w_2') for w_i' in the parent tuple of w_i $(i = 1, 2)$. Then, to generate a trapdoor for a rectangle $R = I_1 \times I_2$, we look for all tuples in the Cartesian product of the prefix decompositions of both intervals I_1, I_2. Note that this implicitly defines a partition in rectangles of R.

Definition IV.2. Let $\mathcal{S}$ be a Boolean searchable symmetric encryption scheme (see Definition II.1). We define a 2-dimensional range searchable symmetric encryption scheme $\mathcal{R}_1$ by means of the following four polynomial-time algorithms:

$\mathcal{R}_1.\mathrm{Gen}(\lambda)$:
Return the output of $\mathcal{S}.\mathrm{Gen}(\lambda)$ and the bit-length $n+1$.

$\mathcal{R}_1.\mathrm{Enc}_k(\mathbf{D})$:
Given a collection of plaintext documents with associated points $\mathbf{D} = ((w_{i,1}, w_{i,2}), D_i)_{i=1}^N$, let $P_{i,j}$ denote the parent tuple of $w_{i,j}$ for $i = 1, \ldots, N$ and $j = 1, 2$.
Let $\mathbf{D}' = \left(\left(w'_{i,1}, w'_{i,2}\right), D_i\right)_{\substack{w'_{i,1} \in P_{i,1}, w'_{i,2} \in P_{i,2} \\ i \in \{1,\ldots,N\}}}$.
Return the output of $\mathcal{S}.\mathrm{Enc}_k(\mathbf{D}')$.

$\mathcal{R}_1.\mathrm{Trpdr}_k(R)$:
Given the rectangle $R = I_1 \times I_2$, denote by Q_j the output of $\mathrm{preDec}(I_i, n+1)$ for $i = 1, 2$. Let $\Phi = \bigvee_{w_1 \in Q_1} \left(w_1 \wedge \left(\bigvee_{w_2 \in Q_2} w_2\right)\right)$ denote the Boolean formula satisfied only by the tuples in $Q_1 \times Q_2$.
Return $T = \mathcal{S}.\mathrm{Trpdr}_k(\Phi)$.

$\mathcal{R}_1.\mathrm{Search}(I, \mathbf{C}, T)$:
Return the output of $\mathcal{S}.\mathrm{Search}(I, \mathbf{C}, T)$.

$\mathcal{R}_1.\mathrm{Dec}_k(C)$:
Return the output of $\mathcal{S}.\mathrm{Dec}_k(C)$.

This scheme can be seen as a two-dimensional adaptation of the scheme defined in [6], where only the one-dimensional case is described.

In the one-dimensional case, the schemes in [6] reveal the length of all prefixes. This speeds up the searching process, but also increases the leakage. We further discuss this issue in Section V.

B. Prefix Decomposition Using Quadtrees

The idea of the previous construction can be also carried out using quadtrees. As we see in Sections V, VI, this has the effect of reducing the leakage incurred by the scheme, while increasing the trapdoor size and search time.

We start by providing an analog to Theorem IV.1 for quadtrees. A proof is given in the full version of this article.

Theorem IV.3. *Let t be a positive integer and let $[a, b] \times [c, d]$ be a rectangle such that both $[a, b]$ and $[c, d]$ have length greater than 2^{t-1} and at most 2^t. Then, it decomposes into at most $7 \cdot 2^{t+1} - 18t - 3$ prefixes.*

A procedure preDec to compute the prefix decomposition of a rectangle $R = [a, b] \times [c, d]$, where $[a, b], [c, d] \in \mathcal{I}_n$ satisfy the restrictions of this theorem, follows directly from the constructive argument in the proof of Theorem IV.3. We describe the algorithm in the full version of this article.

When encrypting a document located at a point (w_1, w_2), the idea behind this second scheme is to attach the document to every element in the parent tuple of $(w_1.w_2)$. Then, to generate a query for a rectangle $R = I_i \times I_2$, we look for all elements in its prefix decomposition.

Definition IV.4. Let $\mathcal{S}$ be a searchable symmetric encryption scheme (see Definition II.1). We define a 2-dimensional range searchable symmetric encryption scheme $\mathcal{R}_2$ by means of the following four polynomial-time algorithms:

$\mathcal{R}_2.\mathrm{Gen}(\lambda)$:
Return the output of $\mathcal{S}.\mathrm{Gen}(\lambda)$ and the bit-length $n+1$.

$\mathcal{R}_2.\mathrm{Enc}_k(\mathbf{D})$:
Given a collection of plaintext documents with associated points $\mathbf{D} = ((w_{i,1}, w_{i,2}), D_i)_{i=1}^N$, let P_i denote the parent tuple of the point $(w_{i,1}, w_{i,2})$.
Let $\mathbf{D}' = (w'_i, D_i)_{\substack{w'_i \in P_i \\ i \in \{1,\ldots,N\}}}$.
Return the output of $\mathcal{S}.\mathrm{Enc}_k(\mathbf{D}')$.

$\mathcal{R}_2.\mathrm{Trpdr}_k(R)$:
Given the rectangle R, denote by Q the output of $\mathrm{preDec}(R, n+1)$. Let $\Phi = \bigvee_{w \in Q} w$ denote the Boolean formula satisfied only by those prefixes in Q.
Return $T = \mathcal{S}.\mathrm{Trpdr}_k(Q)$.

$\mathcal{R}_2.\mathrm{Search}(I, \mathbf{C}, T)$:
Return the output of $\mathcal{S}.\mathrm{Search}(I, \mathbf{C}, T)$.

$\mathcal{R}_2.\mathrm{Dec}_k(C)$:
Return the output of $\mathcal{S}.\mathrm{Dec}_k(C)$.

C. Over-covers Using Binary Trees

There are mainly two drawbacks in using prefix decompositions in binary trees. The first one is that the size of the trapdoor associated to a rectangle R usually increases in the size of the partition of R that the prefix decomposition defines. Consequently, by Theorem IV.1, larger rectangles can have larger trapdoors. The second one is that the size of such trapdoors reveals information about the queried rectangle, since larger trapdoors correspond to larger rectangles.

These concerns are addressed by Faber et al. in [6], and they accordingly propose the concept of over-cover. As defined earlier in Section III-B1, an over-cover of an interval is a set of prefixes such that the union of the associated prefix ranges contains the interval.

As the following theorem states, the relaxation to over-covers allows prefix representations of constant size 3, and so it solves the concerns mentioned in the previous paragraph.

Theorem IV.5 ([6] Theorem 2)**.** *Let t be an integer with $0 \leq t \leq n+1$, and let s, n' be positive integers with $s < t$ and $n' + 2 < 2^s$. Then, any interval $[a, b] \in \mathcal{I}_n$ of length $2^t + 2^s + n' + 2$ admits an over-cover of size 3, with prefixes of length $n - t + 1$, $n - s + 1$ and $\min\{n - s, n - t + 2\}$.*

Nevertheless, Faber et al. note that the interval defined by the corresponding prefix ranges is on average about 40% larger than the original interval (and 66% larger at most), so it may contain values outside the given range. This translates into possibly having false positives in the search results of the scheme.

A simple algorithm to build an over-cover is derived from the proof of Theorem IV.5. We describe the algorithm in the full version of this article.

This third scheme follows the same idea as the $\mathcal{R}_1$ scheme defined in Section IV-A. The difference is that, in the process of generating trapdoors for a rectangle $R = I_1 \times I_2$, we use over-covers of the corresponding intervals instead of prefix decompositions. This implicitly defines a set of rectangles whose union may strictly contain R, and so there can be documents in the result set falling outside of the queried rectangle.

Definition IV.6. Let $\mathcal{S}$ be a Boolean searchable symmetric encryption scheme (see Definition II.1). We define a 2-dimensional range searchable symmetric encryption scheme $\mathcal{R}_3$ by means of the following four polynomial-time algorithms:

$\mathcal{R}_3.\mathrm{Gen}(\lambda)$:
Return the output of $\mathcal{S}.\mathrm{Gen}(\lambda)$ and the bit-length $n+1$.

$\mathcal{R}_3.\mathrm{Enc}_k(\mathbf{D})$:
Given a collection of plaintext documents with associated points $\mathbf{D} = ((w_{i,1}, w_{i,2}), D_i)_{i=1}^N$, let $P_{i,j}$ denote the parent tuple of $w_{i,j}$ for $i = 1, \ldots, N$ and $j = 1, 2$.
Let $\mathbf{D'} = \left(\left(w'_{i,1}, w'_{i,2}\right), D_i\right)_{\substack{w'_{i,1} \in P_{i,1}, w'_{i,2} \in P_{i,2} \\ i \in \{1,\ldots,N\}}}$.
Return the output of $\mathcal{S}.\mathrm{Enc}_k(\mathbf{D'})$.

$\mathcal{R}_3.\mathrm{Trpdr}_k(R)$:
Given the rectangle $R = I_1 \times I_2$, denote by Q_j the output of $\mathrm{preCov}(I_j, n+1)$ for $j = 1, 2$. Let $\Phi = \bigvee_{w_1 \in Q_1} \left(w_1 \wedge \left(\bigvee_{w_2 \in Q_2} w_2\right)\right)$ denote the Boolean formula satisfied only by the tuples in $Q_1 \times Q_2$.
Return $T = \mathcal{S}.\mathrm{Trpdr}_k(\Phi)$.

$\mathcal{R}_3.\mathrm{Search}(I, \mathbf{C}, T)$:
Return the output of $\mathcal{S}.\mathrm{Search}(I, \mathbf{C}, T)$.

$\mathcal{R}_3.\mathrm{Dec}_k(C)$:
Return the output of $\mathcal{S}.\mathrm{Dec}_k(C)$.

D. Over-covers Using Quadtrees

Since, when using quadtrees, the trapdoor size depends on the size of the queried rectangle by Theorem IV.3, both the efficiency and the privacy concerns stated at the beginning of the previous section hold in the $\mathcal{R}_2$ scheme. Thankfully, the concept of over-cover is naturally adapted to quadtrees. As defined earlier in Section III-B2, an over-cover of a rectangle is a set of prefixes such that the union of the associated prefix rectangles contains the rectangle.

As the following theorem states, the relaxation to over-covers allows prefix representations of constant size 13. A proof is given in the full version of this article.

Theorem IV.7. *Let t be an integer with $0 \leq t \leq n-2$ and let $[a,b] \times [c,d]$ be a rectangle such that both $[a,b], [c,d] \in \mathcal{I}_n$ have length at least 2^t, and less than 2^{t+1}. Then, it is contained into the union of 13 prefixes, four of which have length $2(n-t+1)$, and the rest, length $2(n-t+2)$. Also, the area covered by such prefixes is between 56% and 525% larger than the area of the original rectangle.*

In this case, the area covered by the prefixes could be arguably deemed excessive for some applications. Nevertheless, in contrast with the prefix decompositions considered in Sections IV-A,IV-B, the number of used prefixes is always 13, which is constant in t.

A procedure preCov to build an over-cover of a rectangle R is derived from the proof of Theorem IV.7. We describe the algorithm in the full version of this article.

This fourth scheme follows the same idea as the $\mathcal{R}_2$ scheme defined in Section IV-B. The only difference is that, in the process of generating trapdoors for a rectangle $R = [a,b] \times [c,d]$, we use an over-cover of R instead of a prefix decomposition. Also, note that the queried rectangles are restricted so that both intervals have length at least 2^t and less than 2^{t+1} for some integer $0 \leq t \leq n-2$.

As in the previous case, this implicitly defines a set of rectangles whose union strictly contains R, and so there can be documents in the result set falling outside the queried rectangle.

Definition IV.8. Let $\mathcal{S}$ be a searchable symmetric encryption scheme (see Definition II.1). We define a 2-dimensional range searchable symmetric encryption scheme $\mathcal{R}_4$ by means of the following four polynomial-time algorithms:

$\mathcal{R}_4.\mathrm{Gen}(\lambda)$:
Return the output of $\mathcal{S}.\mathrm{Gen}(\lambda)$ and the bit-length $n+1$.

$\mathcal{R}_4.\mathrm{Enc}_k(\mathbf{D})$:
Given a collection of plaintext documents with associated points $\mathbf{D} = (D_i, (w_{i,1}, w_{i,2}))_{i=1}^N$, let P_i denote the parent tuple of the point $(w_{i,1}, w_{i,2})$.
Let $\mathbf{D'} = (w'_i, D_i)_{\substack{w'_i \in P_i \\ i \in \{1,\ldots,N\}}}$.
Return the output of $\mathcal{S}.\mathrm{Enc}_k(\mathbf{D'})$.

$\mathcal{R}_4.\mathrm{Trpdr}_k(R)$:
Given the rectangle $R = [a,b] \times [c,d]$, compute the integers t and t' such that $2^t \leq b-a+1 < 2^{t+1}$ and $2^{t'} \leq d-c+1 < 2^{t'+1}$. If $t \neq t'$ or $t > n-2$, the algorithm halts.
Denote by Q the output of $\mathrm{preCov}(R, n+1)$. Let $\Phi = \bigvee_{w \in Q} w$ denote the Boolean formula satisfied only by those prefixes in Q.
Return $T = \mathcal{S}.\mathrm{Trpdr}_k(Q)$.

$\mathcal{R}_4.\mathrm{Search}(I, \mathbf{C}, T)$:
Return the output of $\mathcal{S}.\mathrm{Search}(I, \mathbf{C}, T)$.

$\mathcal{R}_4.\mathrm{Dec}_k(C)$:
Return the output of $\mathcal{S}.\mathrm{Dec}_k(C)$.

V. Security Analysis

In this section we use the Boolean OXT scheme by Cash et al. [4] in the underlying SSE scheme for our constructions. Simply by following the security analysis done in [4, Section 5.3], we characterize the leakage profile under the *$\mathcal{L}$-semantic-security against adaptive attacks* security definition by Cash et

al. for each of our schemes. We refer the reader to the article [4] for the security definition statement. Also, note that the information leakage of searchable encryption schemes can be reduced by combining them with private information retrieval or oblivious random access memory schemes. In this work we do not discuss these techniques. For more details, see [6].

The *$\mathcal{L}$-semantic-security against adaptive attacks* security definition is stated in the real/ideal paradigm, and it is parametrized by a *leakage function* $\mathcal{L}$. The input of $\mathcal{L}$ is the whole database and all queries exchanged in the game.

If the scheme is $\mathcal{L}$-semantically-secure, $\mathcal{L}$ outputs an upper bound for the leakage incurred by the scheme in the security game. In other words, if the scheme is $\mathcal{L}$-semantically-secure, any probabilistic polynomial-time adversary having access to the whole database and to all queries exchanged in the game is not able to deduce any extra information apart from what it is able to deduce from the output of $\mathcal{L}$. In this way, the output of $\mathcal{L}$ models what is potentially leaked to the server in a protocol execution that follows the security game.

We introduce some terminology and notation in order to describe the output of the leakage function. Suppose that the client uploads the encryption of the collection $((w_{i,1}, w_{i,2}), D_i)_{i=1}^N$ of N plaintext documents. For each plaintext document $((w_1, w_2), D)$ we say that (w_1, w_2) is its *location*, w_1 is its *latitude*, and w_2 is its *longitude*.

Also suppose that, during the game, the client adaptively sends M queries to the server for documents located in rectangles $(R_i)_{i=1}^M$. In the process of generating trapdoors, any such rectangle $R_i = I_{i,1} \times I_{i,2}$ is implicitly expressed as a collection of other rectangles. This is done in different ways depending on the scheme:

- In $\mathcal{R}_1$, we consider every tuple of prefixes in $\mathrm{preDec}(I_{i,1}, n+1) \times \mathrm{preDec}(I_{i,2}, n+1)$.
- In $\mathcal{R}_2$, we consider every prefix in $\mathrm{preDec}(R_i, n+1)$.
- In $\mathcal{R}_3$, we consider every tuple of prefixes in $\mathrm{preCov}(I_{i,1}, n+1) \times \mathrm{preCov}(I_{i,2}, n+1)$.
- In $\mathcal{R}_4$, we consider every prefix in $\mathrm{preCov}(R_i, n+1)$.

Since every prefix expresses a prefix range (in $\mathcal{R}_1$, $\mathcal{R}_3$) or a prefix rectangle (in $\mathcal{R}_2$, $\mathcal{R}_4$), this defines a collection of rectangles. We denote such collection by $(R_{i,1}, \ldots, R_{i,m_i})$ in all cases. Also, for any rectangle $R = I_1 \times I_2$, we say that I_1 is its *latitude* and I_2 is its *longitude*.

We now characterize the leakage profile $\mathcal{L}$ by following the upper bounds on the allowed leakage of the OXT scheme given in [4, Section 5.3].

The output of a leakage function $\mathcal{L}$, such that all of the schemes $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4$ are $\mathcal{L}$-semantically-secure, is determined by:

- Total number of encrypted documents N.
- Results pattern RP: Encrypted documents corresponding to points located in $R_{i,j}$, for each i, j.
- Profile pattern Φ: Whether we are using binary trees or quadtrees, and m_i for all $i = 1, \ldots, M$.

In the cases of the $\mathcal{R}_1$ and $\mathcal{R}_3$ schemes, define the rest of the output of $\mathcal{L}$ by:

- Equality pattern $\bar{s}$: Whether the latitude of two different $R_{i,j}$ coincides.
- Size pattern SP: Number of documents in the latitude of $R_{i,j}$ for each i, j.
- Intersection pattern IP: Encrypted documents whose latitude is contained in the latitudes of two rectangles $R_{i,j}$ having the exact same longitude.

And in the case of the $\mathcal{R}_2$ and $\mathcal{R}_4$ schemes, define the rest of the output of $\mathcal{L}$ by:

- Equality pattern $\bar{s}$: Whether $R_{i,j} = R_{k,l}$ for any i, j, k, l.

Then, as a corollary to the analysis done in [4, Section 5.3], the schemes $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4$ are $\mathcal{L}$-semantically secure against adaptive attacks with $\mathcal{L}$ defined as above.

It is clear that N leaks the same information in all cases. We also deem the information leaked by the results pattern RP as roughly equivalent in all cases, although it is dependent on the expression $(R_{i,1}, \ldots, R_{i,m_i})$ of the queried rectangle R_i.

As to the profile pattern Φ, the situation is presumably very different depending on whether one uses prefix decompositions or over-covers. As observed in [6, Section 3], knowledge about m_i can potentially determine the exact size of the queried rectangle. When using over-covers, m_i is constant in i thanks to Theorems IV.7 and IV.5. Thus, when using over-covers, Φ reveals only whether the scheme uses binary trees or quadtrees. This stronger security guarantee comes at the cost of a high false-positive rate.

As an aside, observe that, if we had included the prefix lengths in the queries (as done for one-dimensional range queries in [6]), then Φ would reveal the exact lengths of the latitude and the longitude of the queried rectangles. Knowledge about prefix lengths would also allow to distinguish between rectangles of the same size. This is fixed in [6, Section 3] by introducing the concept of *universal covers*.

Regarding the equality pattern $\bar{s}$, when using binary trees the leakage seems much larger. The reason is that, for large and sparse enough datasets, knowledge about the equality pattern and the result pattern can be used to decide whether the rectangles in consideration are equal or not. When using quadtrees, $\bar{s}$ is roughly equivalent to having deterministic trapdoors, which is standard leakage in SSE.

Finally, when using binary trees, the size pattern SP and intersection pattern IP leak additional information about documents in the same latitude or longitude than queried rectangles.

VI. Efficiency Analysis

In this section we provide the worst-case complexity analysis of the proposed schemes and of some of the related schemes in the literature. As in the previous section, we use the Boolean OXT scheme by Cash et al. [4] in the underlying SSE scheme for our constructions.

The complexity analysis in Table I is stated in terms of the *resolution* $L = 2^{n+1}$. Thus, L is the number of pixels in each dimension of the grid Λ_n, where the data is located.

Index size is thought as the total size of the index. Trapdoor size and search processing time are taken to be the size of

a trapdoor associated to a rectangle and the time needed to retrieve the corresponding results.

In writing the index size and the search processing time complexity analysis, we assume that every point in the grid Λ_n holds up to one document. This assumption is reasonable since the OXT scheme is suitable for static databases, and plaintext documents are initially thought to be identifiers (see [4, Section 2.1]). More precisely, when outsourcing a dataset, the client encrypts documents of the form $((w_{i,1}, w_{i,2}), \mathrm{id}_i)$ with the SSE scheme, and then associates outsourced ciphertexts to the particular id_i by other means.

Scheme	Index	Trapdoor	Search
SBCSP [10]	$\mathcal{O}(L^2 \log L)$	$\mathcal{O}(\log L)$	$\mathcal{O}(L^2 \log^2(L))$
BW [1]	$\mathcal{O}(L^3)$	$\mathcal{O}(1)$	$\mathcal{O}(L^2)$
$\mathcal{R}_1$	$\mathcal{O}(L^2)$	$\mathcal{O}(\log^2 L)$	$\mathcal{O}(L \log^2 L)$
$\mathcal{R}_2$	$\mathcal{O}(L^2)$	$\mathcal{O}(L \log L)$	$\mathcal{O}(L^3)$
$\mathcal{R}_3$	$\mathcal{O}(L^2)$	$\mathcal{O}(\log L)$	$\mathcal{O}(L)$
$\mathcal{R}_4$	$\mathcal{O}(L^2)$	$\mathcal{O}(\log L)$	$\mathcal{O}(L^2)$

TABLE I
WORST-CASE EFFICIENCY COMPARISON BETWEEN 2-DRSSE SCHEMES.

It is interesting to note that, although in $\mathcal{R}_3$ and $\mathcal{R}_4$ the client issues a constant number of trapdoors per queried rectangle, the size of the trapdoors is logarithmic in L. The use of pseudo-random functions in the trapdoor generation algorithm of OXT scheme forces the size of every trapdoor to scale in $\log L$, and thus it is possible to obtain constant-sized trapdoors by changing the underlying searchable encryption scheme.

VII. Conclusions and Future Work

In this work we propose different techniques for searchable encryption allowing two-dimensional queries on geo-referenced data. We analyze the trade-off between performance, security and communication overhead of the presented options. Some solutions we provide take advantage of the Boolean search and inverted index properties of [4]. We also present a technique based on over-covers that notably reduces the communication cost of the queries at the expense of increasing the false-positive rate.

It is known that, when using searchable encryption, the server can infer statistical information about the stored data. For the case of geo-referenced data, this leakage depends on the election of keywords used to classify the data. It would be interesting to measure the information leakage in the case of range queries, and to find an election of the keywords minimizing it.

As we mention above, we also deal with the case of using this scheme to store geo-referenced data represented by paths and polygons. However, in this case, these documents have many associated points, and so the server could potentially distinguish which documents are attached to single points and which are attached to paths or polygons. A naive solution to prevent this leakage is to store these documents separately.

The use of over-covers is very convenient in our use case, since it substantially reduces the searching time. The drawback of this technique is that the amount of retrieved information is much bigger. The over-covers presented in this work cover a region that can be too big for certain applications, and so we plan to look for a better trade-off between the searching time and the covered region.

A next step in this line of work is to deal with the single-writer multi-reader setting, in which a single client uploads an encrypted dataset and delegates search capabilities to multiple clients. In this case, we are interested in providing access control on the query results. More precisely, we will consider the case where each reader is authorized to search only on data located within a region specified by an access policy. We will need an extension of the searchable encryption scheme supporting access control in the multi-client setting. Also, we will analyze the limitations of the use of over-covers in this setting.

Acknowledgments

This work was supported by the European Comission through H2020-ICT-2014-1-644024 "CLARUS", by the Government of Spain through TIN2014-57364-C2-1-R, and by the Government of Catalonia through Grant 2014 SGR 537. The first author holds a Juan de la Cierva grant.

References

[1] D. Boneh, B .Waters. *Conjunctive, subset, and range queries on encrypted data*. In Proceedings of the 4th conference on Theory of cryptography (TCC'07), Salil P. Vadhan (Ed.). Springer-Verlag, Berlin, Heidelberg, 535–554, 2007.

[2] C. Bösch, P. Hartel, W. Jonker, A. Peter. *A Survey of Provably Secure Searchable Encryption*. ACM Computing Surveys, 47 (2), 18:1–18:51, 2014.

[3] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky. *Searchable symmetric encryption: improved definitions and efficient constructions*. In Proceedings of the 13th ACM conference on Computer and communications security (CCS '06). ACM, New York, NY, USA, 79–88, 2006.

[4] D. Cash, S. Jarecki, C.S. Jutla, H. Krawczyk, M.-C. Rosu, M. Steiner. *Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries*. CRYPTO, 353–373, 2013.

[5] *CLARUS: A framework for user centered privacy and security in the cloud*. Horizon 2020 project H2020-ICT-2014-1-644024. `http://www.clarussecure.eu/`.

[6] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M.-C. Rosu, M. Steiner. *Rich Queries on Encrypted Data: Beyond Exact Matches*. ESORICS, 123–145, 2015.

[7] J. Li, E. R. Omiecinski. *Efficiency and security trade-off in supporting range queries on encrypted databases*. In Proceedings of the 19th annual IFIP WG 11.3 working conference on Data and Applications Security. Springer-Verlag, Berlin, Heidelberg, 69–83, 2005.

[8] M. Naveed, S. Kamara, C. V. Wright. *Inference Attacks on Property-Preserving Encrypted Databases*. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). ACM, New York, NY, USA, 644–655, 2015.

[9] R. A. Popa, C. M. S. Redfield, N. Zeldovich, H. Balakrishnan. *CryptDB: Protecting confidentiality with encrypted query processing*. In Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP-11). 85–100, 2011.

[10] E. Shi, J. Bethencourt, T-H. Hubert Chan, D. Song, A. Perrig. *Multi-Dimensional Range Query over Encrypted Data*. In Proceedings of the 2007 IEEE Symposium on Security and Privacy. IEEE Computer Society, Washington, DC, USA, 350–364, 2007.

[11] D. X. Song, D. Wagner, A. Perrig. *Practical Techniques for Searches on Encrypted Data*. In Proceedings of the 2000 IEEE Symposium on Security and Privacy (SP '00). IEEE Computer Society, Washington, DC, USA, 44–, 2000.