This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2350252, IEEE Transactions on Knowledge and Data Engineering

Authentication of Moving Top-k Spatial Keyword Queries

Dingming Wu, Byron Choi, Jianliang Xu, and Christian S. Jensen

Abstract—A moving top-k spatial keyword (MkSK) query, which takes into account a continuously moving query location, enables a mobile client to be continuously aware of the top-k spatial web objects that best match a query with respect to location and text relevance. The increasing mobile use of the web and the proliferation of geo-positioning render it of interest to consider a scenario where spatial keyword search is outsourced to a separate service provider capable at handling the voluminous spatial web objects available from various sources. A key challenge is that the service provider may return inaccurate or incorrect query results (intentionally or not), e.g., due to cost considerations or invasion of hackers. Therefore, it is attractive to be able to authenticate the query results at the client side. Existing authentication techniques are either inefficient or inapplicable for the kind of query we consider. We propose new authentication data structures, the MIR-tree and MIR*-tree, that enable the authentication of MkSK queries at low computation and communication costs. We design a verification object for authenticating MkSK queries, and we provide algorithms for constructing verification objects and using these for verifying query results. A thorough experimental study on real data shows that the proposed techniques are capable of outperforming two baseline algorithms by orders of magnitude.

Index Terms—Spatial databases, query processing, authentication, spatial keyword search

1 Introduction

A Spatial keyword query [6], [8], [12], [37], [38] integrates location and text search. Taking a location and a set of keywords as arguments, such queries return relevant spatial web objects that match the arguments. Spatial web objects can be points of interest (e.g., restaurants, tourist attractions, hotels, entertainment services) with a web presence, and they thus have locations as well as textual descriptions. A moving top-k spatial keyword (MkSK) query [39], [40], which takes into account a continuously moving query location, enables a mobile client to be continuously aware of the top-k spatial web objects that best match a query with respect to location and text relevance, and it has numerous mobile uses. For example, a mobile client may activate a "cafe" query in order to be alerted about nearby opportunities for a cup of coffee. With the MkSK query, a client always has an up-to-date result as the client moves. The client can ignore a result and just keep moving until an appealing result appears.

A straightforward solution to the MkSK query is to periodically invoke an existing snapshot spatial keyword query processing technique. However, this approach has the problem that even if snapshot queries are processed very frequently, which is expensive and wasteful because consecutive results are likely to be very similar, there is no guarantee that the user always has the right, up-to-date result. Another possible solution is to extend a buffering technique for spatial kNN query processing [35] to top-k spatial keyword querying. Given a kNN query, this technique retrieves $k+\Delta k$ nearest neighbors and uses them to derive a buffer region with the property that as long as the user moves inside the region, the kNN result can be derived from the $k+\Delta k$ objects. However, it is not known how to extend this technique to MkSK queries, where both text relevance and spatial distance are considered. A safe zone based approach [39], [40] for the processing of MkSK queries returns a safe zone to a client together with the query result. The safe zone is a region containing the user's location and in which the top-k result remains unchanged. The safe zone based approach significantly reduces the communication between clients and the service provider as well as computation costs, since the client needs to request new result only when leaving the safe zone.

Example 1.1: Consider the example MkSK query q in Figure 1, where the query keywords are "vanilla coffee." The text relevance of objects p_1 and p_2 to q are 2 and 1, respectively, when using the number of matched keywords for defining relevance. The service provider returns object p_2 as the top-1 result and the gray circle as the safe zone of p_2, meaning that as long as the client remains inside the gray circle, object p_2 is the top-1 result. The curved path shows the client's movement. When the client crosses the boundary of the gray circle (at q'), it sends its updated location to the service provider that then computes and returns a new top-1 result p_1 and the white region as the safe zone. □

- *D. Wu, B. Choi, and J. Xu are with the Department of Computer Science, Hong Kong Baptist University, Hong Kong.*
E-mail: {dmwu,bchoi, xujl}@comp.hkbu.edu.hk
C. S. Jensen is with the Department of Computer Science, Aalborg University, Denmark.
E-mail: csj@cs.aau.dk

This research was supported in part by HK RGC General Research grants (12202414, 210811, and 210510), HKBU Faculty Research grants (FRG/13-14/064 and FRG/12-13/079), and a grant from the Obel Family Foundation.

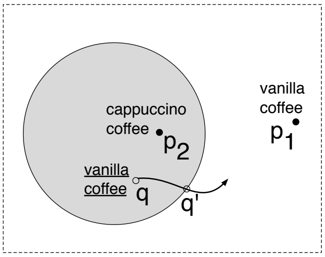


Fig. 1. Example MkSK Query

The web is increasingly being used by mobile users, some 20–50% of web queries have local intent [14], [34], and web content with location continues to proliferate (e.g., web pages, business director entries, tweets, check-ins). This development renders it relevant to enable the outsourcing of spatial keyword querying. Such outsourcing of query functionality by an entity that wishes to deliver a service, called a data owner (DO), to a service provider (SP) offers a number of advantages [15], including elastic scaling, cost savings, and high availability when compared to conventional client-server architectures. Moreover, data owners need not buy high performance hardware and hire technical staff to maintain the service. The outsourcing is likely to also improve service availability. In addition, outsourcing can reduce the network latency associated with the delivery of query results.

However, being outside of the administrative scope of the DO, the SP may return inaccurate or incorrect query results, intentionally or unintentionally, e.g., due to cost considerations or invasion of hackers [36]. In our setting, the SP may return an overly large safe zone together with a query result in order to save computing resources and communication bandwidth [27], [43]. The SP may also return an overly small safe zone to incur frequent requests from the client if the SP is paid per request. Therefore, *query authentication*, i.e., establishing the soundness and completeness of query results at the client side, is highly desirable. *Soundness* guarantees that every result object is present in the DO's database and is not tampered with. *Completeness* ensures that no valid result object is missed in a query result.

Authentication techniques have been developed for a variety of queries, including relational queries [19], [26], [41], sliding window queries [20], [29], spatial queries [16], [42], text similarity queries [27], shortest path queries [43], moving kNN queries [44], moving range queries [45], and subgraph search [11], [30]. However, all existing Authenticated Data Structures (ADS) are either inapplicable or inefficient in relation to our setting, since the authentication of MkSK queries involves verifying both spatial proximity and text relevance. Moreover, authenticating an MkSK query includes verifying both the top-k result and the accompanying safe zone. The safe zone is calculated based on both the objects in the top-k result and the objects not in the top-k result, so that missing a non-result object may cause a safe zone to fail in the authentication. Although authentication techniques for moving kNN queries [44] and moving range queries [45] involve safe zone verification, the safe zone of an MkSK query is very different, i.e., it is a Multiplicatively Weighted (MW) Voronoi cell [25].

To the best of our knowledge, this paper is the first to study the authentication of moving top-k spatial keyword queries. Our proposals also apply to the authentication of static top-k spatial keyword queries (only verifying the top-k result), which is a sub-task of the authentication of MkSK queries. Our contributions include:

- We design a new data structure for the authentication of MkSK queries, the MIR-tree, that enables low computation and communication costs. The MIR-tree is applicable to many variant of the spatial keyword query.
- We design a Verification Object ($\mathcal{VO}$) for authenticating the top-k results and safe zones of MkSK queries. Algorithms for constructing $\mathcal{VO}$ and verifying the top-k results and safe zones using $\mathcal{VO}$ are developed.
- An enhanced data structure, the MIR*-tree, is proposed to further reduce the communication cost. The idea of the MIR*-tree is applicable for any tree-structured ADS, e.g., the MR-tree [42], where each node contains multiple entries.
- We conduct a thorough experimental study on real data to evaluate the performance of our proposals. The proposed approaches are able to outperform two baseline algorithms exploiting existing authentication techniques by orders of magnitude.

The rest of the paper is organized as follows. The problem setting is presented in Section 2. The two new ADSs, the MIR-tree and the MIR*-tree, the design of a $\mathcal{VO}$, and accompanying algorithms for the authentication of MkSK queries are introduced in Section 3. We describe the two baseline algorithms and study the performance of our proposals in Section 4. Related work is covered in Section 5, and we conclude in Section 6.

2 Problem Setting

2.1 Moving Top-k Spatial Keyword Query

We consider a generic problem setting in which the future locations of an MkSK query cannot be predicted in advance [39], [40]. Let $\mathcal{D}$ be a data set in which each object $p \in \mathcal{D}$ is a pair $\langle \lambda, \psi \rangle$ of a point location $p.\lambda$ and a text description, or document, $p.\psi$ (e.g., the facilities and menu of a restaurant). A moving top-k spatial keyword (MkSK) query $q = \langle \lambda, \psi, k \rangle$ takes three arguments: a continuously changing point location λ, a set of keywords ψ, and a number of requested objects k. Intuitively, an object whose description is more relevant to the query keywords and is closer to the query location is preferable. We use a weighted distance (Equation 1) [3] as our ranking function. This type of function has been used extensively in the geosciences [13], [18] and it matches the semantics of the query. A key strength of this function is that the different units of measurement of the spatial distance and textual relevance is canceled out in the ratio and thus do not affect the result.

$$rank_q(p) = \frac{\|q\ p\|}{tr_q(p)}, \tag{1}$$

where $\|q\ p\|$ denotes their Euclidean distance and $tr_q(p)$ denote the text relevancy of p to q. The MkSK query retrieves k objects $\mathcal{RS}$ that minimize the ranking value and a safe zone $\Upsilon^k(\mathcal{RS})$ within which the top-k result is valid. Formally, $\forall p \in \mathcal{RS}\ (\forall p' \in \mathcal{D} - \mathcal{RS}\ (rank_q(p) \leq rank_q(p')))$ and $q.\lambda \in \Upsilon^k(\mathcal{RS}) \implies \mathcal{RS}$ is the top-k result.

The safe zone of the top-k result of an MkSK query is an order-k MW-Voronoi cell [25] that is an irregular geometric entity. The text relevancy of an object to the query is the weight used in the computation of MW-Voronoi cells. We thus use weight $w(\cdot)$ and text relevancy $tr_q(\cdot)$ interchangeably. Note that the safe zone preserves the set of top-k objects, not the ranking of the top-k objects. More details about safe zones is provided in Appendix A. Table 6 in Appendix A summarises the symbols used in the paper.

2.2 Authentication Framework

Figure 2 illustrates the framework for answering MkSK queries that supports correctness verification. The framework consists of two phases, i.e., initialization and query processing & authentication. In the initialization phase, the DO first gets a private key from a key distribution center. Next, it signs the ADS constructed on the data set using the private key and transfers the ADS and signatures to the SP. A client downloads a public key from the key distribution center and the signatures from the SP. In the query processing and authentication phase, the client first issues an MkSK query. Upon receiving the query, the SP computes the top-k result, the safe zone, and a verification object ($\mathcal{VO}$) that encodes the query result and its safe zone. The client gets the $\mathcal{VO}$ from the SP. The top-k result $\mathcal{RS}$ and its safe zone $\Upsilon^k(\mathcal{RS})$ are obtained from the $\mathcal{VO}$. The correctness of the top-k result and the safe zone can be verified by the client using the $\mathcal{VO}$, the signatures, and the public key. The client needs to send a new request to the SP only when it leaves the safe zone.

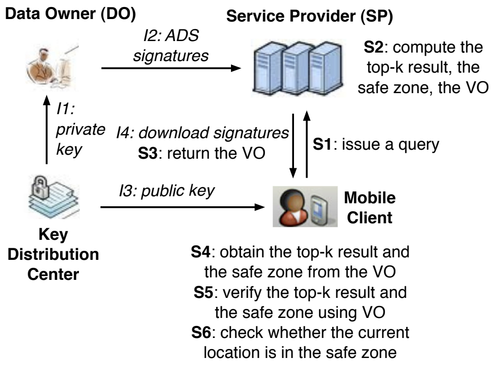

Data Owner (DO)
Service Provider (SP)
I2: ADS signatures
S2: compute the top-k result, the safe zone, the VO
I1: private key
I4: download signatures
S3: return the VO
S1: issue a query
I3: public key
Mobile Client
Key Distribution Center
S4: obtain the top-k result and the safe zone from the VO
S5: verify the top-k result and the safe zone using VO
S6: check whether the current location is in the safe zone

Fig. 2. MkSK Query Authentication

2.3 Authentication Tasks

The authentication of an MkSK query q amounts to guaranteeing that the client always has the correct top-k result while its spatial location changes continuously. The safe zone based approach guarantees that as long as the query does not exit the safe zone, the received top-k result remains valid. In other words, when the query moves across the boundary of a safe zone, it requests an updated top-k result and corresponding safe zone. Therefore, authenticating an MkSK query is equivalent to verifying the correctness of both the top-k result $\mathcal{RS}$ and the corresponding safe zone. The sufficient condition of MkSK query authentication can be further divided into four sub-conditions.

(I) Soundness of a top-k result.

$$\forall p \in \mathcal{RS}\ (p \in \mathcal{D}) \tag{2}$$

Every object in the top-k result must present in DO's data set. Specifically, both the location and the text description of the object are not tampered with and the ranking score $rank_q(p)$ of p is computed correctly.

(II) Completeness of a top-k result.

$$\forall p \in \mathcal{RS}\ (\forall p' \in \mathcal{D} - \mathcal{RS}\ (rank_q(p) \leq rank_q(p'))) \tag{3}$$

All objects not in the top-k result have ranking scores that are no better than any of those of the result objects.

(III) Soundness of a safe zone.

$$\forall p \in I\ (p \in \mathcal{D}) \tag{4}$$

Since a safe zone is computed from the result objects and the influence objects [40], the facts that the locations and the text descriptions of the influence objects are not tampered with, the text relevancies of influence objects are computed correctly, and sub-condition (I) guarantee the soundness of a safe zone.

(IV) Completeness of safe zone.

$$\forall \lambda' \in \Upsilon^k(\mathcal{RS})\ (Q(\lambda', \psi, k) = Q(\lambda, \psi, k) \bigwedge \\ \forall \lambda' \notin \Upsilon^k(\mathcal{RS})\ (Q(\lambda', \psi, k) \neq Q(\lambda, \psi, k) \tag{5}$$

For all locations λ' in the safe zone, the result of the query with parameters λ', the keywords of the original query ψ, and k on the original data set is the same as the result of the original query $Q(\lambda, \psi, k)$. For all λ' not in the safe zone, the two results are different.

2.4 Threat Model

The SP is the potential adversary [42]. The SP is outside the administrative scope of the DO and thus cannot be trusted. With the exception of the DO's private key, adversaries are assumed to know all information, including the public key for the secure-hash function, the ADS, the signatures, and the authentication algorithms. They may alter the data set or the ADS, and they may tamper with the search result. Issues related to privacy are beyond the scope of this paper.

2.5 Problem Statement

We study the authentication of MkSK queries, aiming for a solution that (i) verifies the soundness and completeness of the top-k result and the safe zone of a given MkSK query and (ii) optimizes the server-side computational cost and the client/server communication cost, i.e., the $\mathcal{VO}$ size.

3 Proposed Solution

Most existing authentication techniques construct a region (range), also called a *verification set*, that covers the query result. The data objects that fall into the verification set are sent to the client. In addition, summary objects that are used to derive bounds on the ranking scores of the objects outside the verification set are sent to the client. In other words, the $\mathcal{VO}$ contains the objects inside the verification set and a number of summary objects. Using this $\mathcal{VO}$, the client is able to authenticate the result. Since the result of an MkSK query is retrieved according to Equation 1, authenticating an MkSK query involves verifying both the spatial distances and text relevancies of the result objects. The current state-of-the-art solution for authenticating spatial queries, the MR-tree [42], and the existing method for authenticating the query results of text search engines [27] can be used to authenticate the spatial and textual parts of the result objects separately. However, this approach is inefficient because it generates a large $\mathcal{VO}$ based on a large verification set, which causes high communication cost and long authentication time, as we will show in the paper's experimental study. We propose a new ADS, the MIR-tree (Section 3.1), for efficiently verifying MkSK queries. It is also applicable to spatial keyword queries based on other ranking functions. The proposed authentication algorithms (in Sections 3.2 and 3.3) construct a compact $\mathcal{VO}$, i.e., put minimum numbers of objects and MIR-tree entries into the $\mathcal{VO}$. Hence, the communication cost and authentication time are saved. To further reduce the size of a $\mathcal{VO}$, we propose an enhanced ADS, the MIR*-tree (in Section 3.4).

3.1 MIR-Tree

The Merkle-IR-tree (MIR-tree) is developed based on the IR-tree [8], by embedding a series of digests in each node of the tree. In the IR-tree, each entry summarizes the spatial distances and text relevancies of the entries in its child node. This index enables the efficient processing of spatial keyword queries, since it is able to prune the search space according to spatial proximity and text relevance simultaneously. In the MIR-tree, to authenticate text relevancies, a *word digest* is stored with each entry in each posting list in the inverted file attached to each non-leaf node. Formally, for a word w, a posting list entry takes the form $(id, weight, h_w(e))$, where id is the identifier of an entry e in a node in the tree, $weight$ is the weight of w in the pseudo text description of e, and $h_w(e)$ is the word digest of e for word w. The word digest of e for w is computed from the hash value of the concatenation '|' of the binary representation of all the entries in the posting list of w in the child node of e:

$$h_w(e) = h((ce_1, weight, h_w(ce_1))| \\ (ce_2, weight, h_w(ce_2))| \cdots), \quad (6)$$

where ce_i $(i = 1, 2, \cdots)$ is an entry in the child node of e and $h()$ is a secure hash function. Entries in leaf nodes do not have word digests.

Figure 3 and Table 1 illustrate how word digests are stored and computed, respectively. For example, in $InvF$-R_5 the word digest of word 'c' ($h_c(R_1)$) for non-leaf entry R_1 is computed as $h_c(R_1) = h((p_1, 5)|(p_2, 5))$; in $InvF$-*root* the word digest of word 'c' ($h_c(R_5)$) for non-leaf entry R_5 is computed as $h_c(R_5) = h((R_2, 7, h_c(R_2))|(R_1, 5, h_c(R_1)))$, which takes the digests in the child nodes ($h_c(R_1)$ and $h_c(R_2)$) into account. The digests of other words are computed similarly. Note that, word digests do not occupy any space in nodes. They are stored in the inverted files attached to nodes. The word digests of the root node (e.g., $h_c(root)$) are signed by the data owner (e.g., $RSA(h_c(root))$) and are stored with the MIR-tree. Each word has a root signature.

Furthermore, to authenticate spatial distances, a *spatial digest* is stored with each non-leaf entry. Formally, a non-leaf node contains entries of the form $(R_i, R_i.\Lambda, R_i.h)$, where R_i is a reference to its child node, $R_i.\Lambda$ is the MBR of all rectangles in entries of the child node, and $R_i.h$ is the spatial digest of the child node. The leaf nodes are identical to those of the IR-tree. The spatial digest of a leaf node is computed from the hash value of the concatenation of the binary representation of all objects in the node. The spatial digest of a non-leaf node summarizes its child nodes' MBRs as well as their digests. The root spatial digest is signed and stored with the MIR-tree. Figure 3 and Table 2 illustrate how spatial digests are stored and computed, respectively.

The shaded parts in Figure 3 show the additions to the MIR-tree over the IR-tree. The update cost of the MIR-tree is comparable to that of the IR-tree. Specifically, when inserting or deleting an object in a leaf node, the spatial and word digests in its parent node are re-computed only if the MBR and the word weights of the parent node are changed. If splitting happens, the spatial and word digests of affected nodes are re-computed. Changes are propagated to the root node.

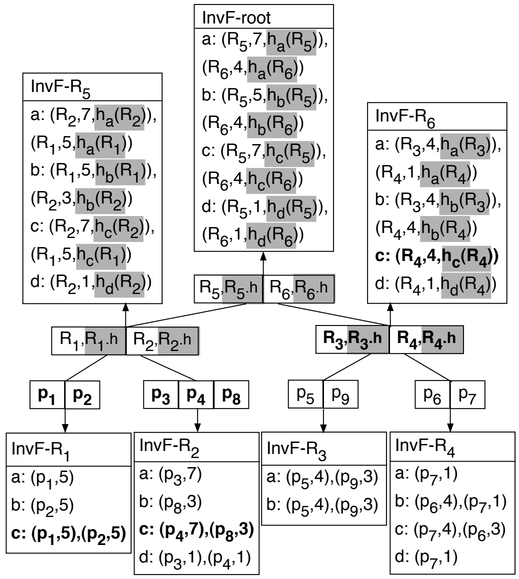


Fig. 3. Example MIR-Tree

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2350252, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Example Word Digest Computations

InvF-R_5	$h_a(R_2) = h((p_3, 7))$
	$h_a(R_1) = h((p_1, 5))$
	$h_b(R_1) = h((p_2, 5))$
	$h_b(R_2) = h((p_8, 3))$
	$h_c(R_2) = h((p_4, 7)\|(p_8, 3))$
	$h_c(R_1) = h((p_1, 5)\|(p_2, 5))$
	$h_d(R_2) = h((p_3, 1)\|(p_4, 1))$
InvF-R_6	$h_a(R_3) = h((p_5, 4)\|(p_9, 3))$
	$h_a(R_4) = h((p_7, 1))$
	$h_b(R_3) = h((p_5, 4)\|(p_9, 3))$
	$h_b(R_4) = h((p_6, 4)\|(p_7, 1))$
	$h_c(R_4) = h((p_7, 4)\|(p_6, 3))$
	$h_d(R_4) = h((p_7, 1))$
InvF-root	$h_a(R_5) = h((7, h_a(R_2))\|(5, h_a(R_1)))$
	$h_a(R_6) = h((4, h_a(R_3))\|(1, h_a(R_4)))$
	$h_b(R_5) = h((5, h_b(R_1))\|(3, h_b(R_2)))$
	$h_b(R_6) = h((4, h_b(R_3))\|(4, h_b(R_4)))$
	$h_c(R_5) = h((7, h_c(R_2))\|(5, h_c(R_1)))$
	$h_c(R_6) = h((4, h_c(R_4)))$
	$h_d(R_5) = h((1, h_d(R_2)))$
	$h_d(R_6) = h((1, h_d(R_4)))$
	$h_a(root) = h((7, h_a(R_5))\|(4, h_a(R_6)))$
	$h_b(root) = h((5, h_b(R_5))\|(4, h_b(R_6)))$
	$h_c(root) = h((7, h_c(R_5))\|(4, h_c(R_6)))$
	$h_d(root) = h((1, h_d(R_5))\|(1, h_d(R_6)))$

TABLE 2
Example Spatial Digest Computations

$R_1.h = h(p_1\|p_2)$
$R_2.h = h(p_3\|p_4\|p_8)$
$R_3.h = h(p_5\|p_9)$
$R_4.h = h(p_6\|p_7)$
$R_5.h = h(R_1.\Lambda\|R_1.h\|R_2.\Lambda\|R_2.h)$
$R_6.h = h(R_3.\Lambda\|R_3.h\|R_4.\Lambda\|R_4.h)$
$root.h = h(R_5.\Lambda\|R_5.h\|R_6.\Lambda\|R_6.h)$

3.2 Top-k Result Authentication

The challenge of verifying a top-k result involves the design of a compact $\mathcal{VO}$ that achieves low communication cost and short authentication time. The inefficiency of authenticating spatial distances and text relevancies separately using existing techniques [27], [42] can be illustrated by the following example. Imagine a data set of 5 objects $p_1, \ldots, p_5$ having spatial distances 1, 2, 3, 4, 5 and text relevancies 5, 6, 7, 8, 20, with regard to a query. According to the ranking function in Equation 1, objects p_1 and p_5 make up the top-2 result with ranking scores 0.2 and 0.25. If authenticating the spatial distance and text relevance separately, one verification set covering the objects with spatial distances smaller than 5 (the maximum spatial distance in the result) and the other verification set covering the objects with text relevancies larger than 5 (the minimum text relevance in the result) are constructed. Therefore, the whole data set is covered by the two verification sets and added to the $\mathcal{VO}$, which incurs very high communication cost. Idealy, only one verification set covering the objects (p_1 and p_5) with ranking scores smaller than 0.25 (the maximum value of a ranking score of a result object) should be constructed. Thus, only three objects p_1, p_5 and a summary object representing all the other objects in the data set make up a $\mathcal{VO}$ that achieves low communication cost.

3.2.1 $\mathcal{VO}$ Construction

The proposed ADS, the MIR-tree, has the desirable property that only a minimum number of objects and MIR-tree entries need to be inserted into the $\mathcal{VO}$. We construct a verification set VS_{rs} covering the objects with ranking scores smaller than $rank_k = \max_{p \in \mathcal{RS}}\{rank_q(p)\}$, i.e., the maximum score in the top-k result. Recall that the smaller the score, the better the ranking. Algorithm 1 shows the pseudo code for constructing the $\mathcal{VO}$ for the top-k result. Each entry e in the $\mathcal{VO}$ takes the form $(id, \Lambda, H, \{(w, weight, h_w(\cdot))\})$, where id is an identifier, Λ is an MBR, H is a spatial digest, and $(w, weight, h_w(\cdot))$ is a triple where w is the identifier of one query keyword, $weight$ is the weight of word w, and $h_w(\cdot)$ is the word digest. If id refers to an object, its H and $h_w(\cdot)$ values are empty.

The $\mathcal{VO}$ is computed by a depth-first traversal of the MIR-tree using the following conditions: (i) if a non-leaf entry e has a ranking score $rank_q(e) = \|q\ e\|_{min}/tr_q(e)$ that is higher than $rank_k$, the $\mathcal{VO}$ entry for e is constructed and added to the $\mathcal{VO}$, and its subtree will not be visited (lines 7–9); otherwise, Algorithm 1 is called to process the child node of e (lines 25–26); (ii) for any visited leaf node, all the objects in it are added to the $\mathcal{VO}$ (line 7). When constructing the $\mathcal{VO}$, a traversal string str_{rs} is composed that tracks the search in the MIR-tree. It contains the identifiers of the tree entries and objects added to the $\mathcal{VO}$, as well as the special tokens '[' and ']' used to mark the scope of a node (line 11). The traversal string is needed in order to avoid having duplicate entries in the $\mathcal{VO}$, since the $\mathcal{VO}$ for the top-k result and the $\mathcal{VO}$ for the safe zone may otherwise have common entries, as will be explained later. Finally, the $\mathcal{VO}$ and the traversal string str_{rs} are sent to the client.

Algorithm 1 VOConstruction (MIRtreeNode N, double $rank_k$, MkSKQuery q)

```
1: Append [ to $str_{rs}$;
2: for each entry $e$ in $N$ do
3:     if $N$ is a leaf node then
4:         Add $VOEntry(e)$ to $\mathcal{VO}$;
5:         Append $e$ to $str_{rs}$;
6:     else                                  ▷ $N$ is a non-leaf node.
7:         if $rank_q(e) > rank_k$ then
8:             Append $e$ to $str_{rs}$;
9:             Add $VOEntry(e)$ to $\mathcal{VO}$;
10:        else
11:            VOConstruction ($e$, $rank_k$, $q$);
12: Append ] to $str_{rs}$;
13: Return $\mathcal{VO}$ and $str_{rs}$;
```

Example 3.1: Figure 3 shows the MIR-tree built on the spatial web objects in Figure 4. Let q be an MkSK query with query keyword 'c', where $k = 1$. Its top-1 result is object p_1. The traversal string str_{rs} is $[[[p_1p_2][p_3p_4p_8]][R_3R_4]]$. The $\mathcal{VO}$ is $\{(p_1, p_1.\Lambda, \{(c, 5)\})$, $(p_2, p_2.\Lambda, \{(c, 5)\})$, $(p_3, p_3.\Lambda, \{(c, 7)\})$, $(p_4, p_4.\Lambda, \{\})$, $(p_8, p_8.\Lambda, \{(c, 3)\})$, $(R_3, R_3.\Lambda, R_3.H, \{\})$, $(R_4, R_4.\Lambda, R_4.H,$ $\{(c, 4, h_c(R_4))\})\}$. The entries shown using bold font in Figure 3 are added to the $\mathcal{VO}$. Since object p_4 and entry

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2350252, IEEE Transactions on Knowledge and Data Engineering

R_3 do not contain word 'c', their $(w, weight, h_w(\cdot))$ sets are empty. The root spatial signature RSA $(root.H)$ and the root word signature $RSA(h_c(root))$ are sent to the client together with the $\mathcal{VO}$ and the traversal string. □

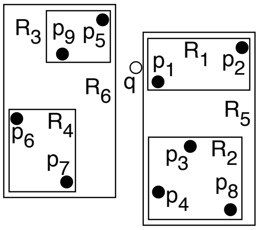


Fig. 4. Spatial Web Objects

3.2.2 Verifying a Top-k Result

The client extracts the top-k result from the $\mathcal{VO}$ and authenticates it. It first computes the ranking scores of the objects in the $\mathcal{VO}$ using Equation 1, ranks them, and obtains the top-k objects. Next, the client re-computes the root spatial and word digests using the str_{rs} and $\mathcal{VO}$ (Algorithm 2) and verifies them against the decrypted root spatial and word signatures. The main idea is to re-construct the MIR-tree traversal performed by the server, i.e., guaranteeing the entries in the $\mathcal{VO}$ are from the original MIR-tree. Then, it verifies the correctness of the top-k result by checking whether the ranking score computed from Equation 1 of the k^{th} object is no worse than those of all the other entries in the $\mathcal{VO}$.

Example 3.2: Consider the traversal string str_{rs} and the $\mathcal{VO}$ in Example 3.1. The root spatial digest is re-constructed as $h(MBR(p_1.\Lambda, p_2.\Lambda, p_3.\Lambda, p_4.\Lambda, p_8.\Lambda)| h(MBR(p_1.\Lambda, p_2.\Lambda)|h(p_1|p_1.\Lambda|p_2|p_2.\Lambda)|MBR(p_3.\Lambda, p_4.\Lambda, p_8.\Lambda)|h(p_3|p_3.\Lambda|p_4|p_4.\Lambda|p_8|p_8.\Lambda))|MBR(R_3.\Lambda, R_4.\Lambda)| h(R_3.\Lambda|R_3.H|R_4.\Lambda|R_4.H))$. The root word digest of 'c' is $h((7, h((7, h((p_4, 7)|(p_8, 3)))|(5, h((p_1, 5)|(p_2, 5)))))| (4, h((4, d_c(R_4)))))$. The steps needed to compute the spatial and word digests are shown in Figure 5. □

3.2.3 Soundness and Completeness

In this section, we prove that our proposal can authenticate the top-k result.

Proof of Soundness.

Suppose that an object p in the result is fake or modified. The re-constructed root spatial and word digests (hash values) cannot be verified against the signatures provided by the data owner, which is detected, since the hash function is one way collision-resistant and p and its word weights must be used by Algorithm 2.

Proof of Completeness.

Let p be one of the top-k objects and let N be the leaf node containing p. If all the entries in N are in the $\mathcal{VO}$, for sure p is in the $\mathcal{VO}$ and is included in the top-k result. If N is pruned by the server, the MBR, keyword weights and digests of N are in the $\mathcal{VO}$. The client obtains the top-k result from the $\mathcal{VO}$, which does not include p. Since the re-constructed root spatial and word digests (hash values) equal the decrypted signatures, the ranking score of N is correct and must be better than that of the k^{th} object, which informs the client about a potential completeness violation.

Algorithm 2 (MBR, SpatialDigest, WordWeightList, WordDigestList) **Authentication** (VerificationObject $\mathcal{VO}$, String str_{rs}, MkSKQuery q)

```
$spatialstr \leftarrow null$;
$wordstrlist \leftarrow null$;
$MBR \leftarrow null$;
$WordWeightList \leftarrow null$;
$WordDigestList \leftarrow null$;
for each word $w$ in the query keywords do
    $w.str \leftarrow null$;
    $wordstrlist.add(w.str)$;
    $w.weight \leftarrow -1$;
    $WordWeightList.add(w.weight)$;
    $w.digest \leftarrow 0$;
    $WordDigestList.add(w.digest)$;
while $str_{rs}$ is not empty do
    $e_V \leftarrow str_{rs}.nextEntry()$;
    if $e_V$ refers to an object then
        $p \leftarrow \mathcal{VO}.getEntry(e_V)$;
        $MBR \leftarrow MBR \cup p.\Lambda$;
        $spatialstr \leftarrow spatialstr|p.id|p.\Lambda$;
        for each word $w'$ in $\{(w, weight, h_{w'}(\cdot))\}$ do
            Find the corresponding $w$ in $wordstrlist$ and $WordWeightList$;
            $w.str \leftarrow w.str|(p.id, w'.weight)$;
            $w.weight \leftarrow \max(w.weight, w'.weight)$;
    if $e_V$ refers to a non-leaf entry in the MIR-tree then
        $e \leftarrow \mathcal{VO}.getEntry(e_V)$;
        $MBR \leftarrow MBR \cup e.\Lambda$;
        $spatialstr \leftarrow spatialstr|e.\Lambda|e.H$;
        for each word $w'$ in $\{(w, weight, h_{w'}(\cdot))\}$ do
            Find the corresponding $w$ in $wordstrlist$ and $WordWeightList$;
            $w.str \leftarrow w.str|(w'.weight, h_{w'}(\cdot))$;
            $w.weight \leftarrow \max(w.weight, w'.weight)$;
    if $e_V$ is [ then
        $(mbr, sd, wwl, wdl) \leftarrow$ Authentication ($\mathcal{VO}$, $str_{rs}$, $q$);
        $MBR \leftarrow MBR \cup mbr$;
        $spatialstr \leftarrow spatialstr|mbr|sd$;
        for each word $w'$ in $wwl$ and $wdl$ do
            Find the corresponding $w$ in $wordstrlist$ and $WordWeightList$;
            $w.str \leftarrow w.str|(w'.weight, h_{w'}(\cdot))$;
            $w.weight \leftarrow \max(w.weight, w'.weight)$;
    if $e_V$ is ] then
        for each word $w$ in $WordDigestList$ and $wordstrlist$ do
            $w.digest \leftarrow h(w.str)$;
        Return $(MBR, h(spatialstr), WordWeightList,$
$WordDigestList)$;
```

3.3 Safe Zone Authentication

Recall that a safe zone is defined by influence objects. The challenge of verifying the safe zone is to ensure that the client notices that influence objects are missing if the SP omits some influence objects. Considering the safe zone in Figure 6(a), suppose that the SP only sends influence objects p_2 and p_4 to the client, omitting p_3. Then the client will construct an incorrect (larger) safe zone, namely the gray region in Figure 6(b). The influence objects p_2 and p_4 are originally from the data set that can pass the verification

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2350252, IEEE Transactions on Knowledge and Data Engineering

by checking root signatures. However, the client cannot determine whether the influence object set provided by the SP is complete.

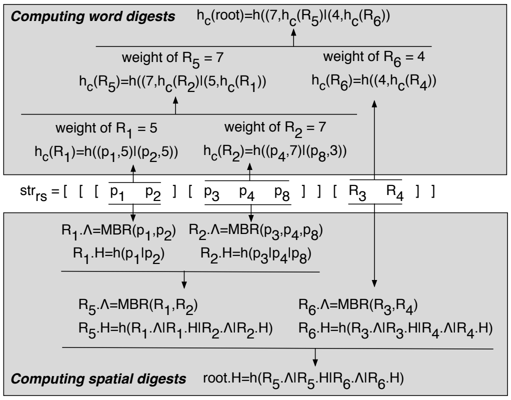
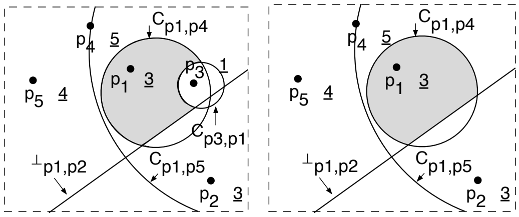


Fig. 5. Computation of Spatial and Word Digests



(a) Safe Zone (b) Fake Safe Zone

Fig. 6. Example Safe Zones

A naive approach is to send the whole data set to the client so that the client is guaranteed to compute the correct safe zone. However, this involves the high communication cost of transferring the whole data set and is impractical since the client may have neither enough storage to store the data set nor computational resources to efficiently compute the result. We aim to design a $\mathcal{VO}$ for authenticating the safe zone such that the $\mathcal{VO}$ is as small as possible.

3.3.1 $\mathcal{VO}$ Construction

We define a verification set VS_{sz} that is intended to contain the objects that are useful for verifying the safe zone, and we insert the objects in this region into the $\mathcal{VO}$. Verification set VS_{sz} is calculated based on the **minimum** and **maximum border distances** [40] defined as follows:

Definition 3.1: The minimum border distance of a non-result object p from a result object p^* is the minimum distance between p^* and the border of its dominant region $Dom_{p^*,p}$, i.e.,

$$bord_{min}(p^*, p) = bord_{min}(p^*, Dom_{p^*,p}) = \|p^*\ p\| \frac{w(p^*)}{w(p^*) + w(p)}. \quad (7)$$

Definition 3.2: The maximum border distance of a non-result object p from a result object p^* is the maximum distance between p^* and the border of its dominant region $Dom_{p^*,p}$, i.e.,

$$bord_{max}(p^*, p) = bord_{max}(p^*, Dom_{p^*,p}) = \|p^*\ p\| \frac{w(p^*)}{|w(p^*) - w(p)|}. \quad (8)$$

We first consider the safe zone of a top-1 object p^*. Let I^+ be the set of influence objects of the safe zone of p^*, where the weight of each influence object exceeds that of p^*. A threshold τ is defined as the minimum value of the maximum border distances of the influence objects in I^+, i.e.,

$$\tau = \min_{p \in I^+} bord_{max}(p^*, p). \quad (9)$$

Verification set VS_{sz} of the safe zone of p^* covers the objects whose minimum border distances are smaller than τ.

As an example, the gray region in Figure 7 is the safe zone of top-1 object p^*. The influence object set I^+ contains p_1 and p_4. Threshold $\tau = bord_{max}(p^*, C_{p^*,p_1})$ is shown as the dashed circle. The verification set VS_{sz} of the safe zone covers objects p_1 and p_2 that are influence objects.

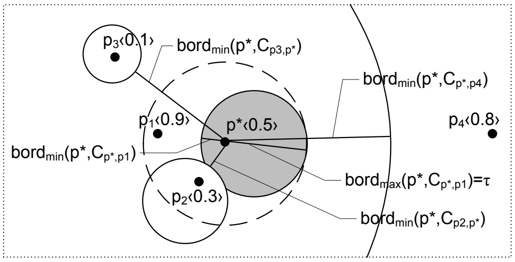


Fig. 7. Example Safe Zone and Influence Objects

We proceed to define the verification set of the safe zone of top-k result $\mathcal{RS}$. The order-k MW-Voronoi cell of set $\mathcal{RS}$, denoted by $\Upsilon^k(\mathcal{RS})$, contains all locations that take set $\mathcal{RS}$ as the top-k result. In other words, $\Upsilon^k(\mathcal{RS})$ is the safe zone for the result set $\mathcal{RS}$. Region $\Upsilon^k(\mathcal{RS})$ can be represented by the intersections of order-1 MW-Voronoi cells [25]:

$$\Upsilon^k(\mathcal{RS}) = \bigcap_{p_j^* \in \mathcal{RS}} \Upsilon_{\mathcal{D}-\mathcal{RS}}(p_j^*), \quad (10)$$

where $\Upsilon_{\mathcal{D}-\mathcal{RS}}(p_j^*)$ denotes the MW-Voronoi cell of p_j^* with respect to the object set $\mathcal{D} - \mathcal{RS}$. For each result object p_j^*, threshold τ_j is computed as $\tau_j = \min_{p \in I_j^+} bord_{max}(p_j^*, p)$, where I_j^+ is the influence object set of p_j^*. We next give the formal definition of the verification set of the safe zone.

Definition 3.3: The verification set VS_{sz} of a safe zone of top-k result $\mathcal{RS}$ covers the objects that satisfy the following condition:

$$\bigwedge_{p_j^* \in \mathcal{RS}} bord_{min}(p_j^*, p) \le \tau_j \quad (11)$$

Theorem 3.1: The objects covered by the verification set VS_{sz} of the safe zone of top-k result $\mathcal{RS}$ are sufficient to re-construct the safe zone.

Proof: The theorem implies that the objects outside the verification region do not contribute the safe zone. According to Definition 3.3, any object p outside the verification set VS_{sz} satisfies the following condition:

$$\bigvee_{p_j^* \in \mathcal{RS}} bord_{min}(p_j^*, p) > \tau_j \quad (12)$$

Suppose the minimum border distance of p exceeds the threshold τ of one result object p^*, i.e., $bord_{min}(p^*, p) > \tau$. Based on the definition of the minimum and maximum border distances (Definitions 3.1 and 3.2) and τ (Equation 9), the dominant region $Dom_{p^*,p}$ completely contains the safe zone of p^*, i.e., $Dom_{p^*,p} \supset \Upsilon(p^*)$. In other words, object p does not contribute to the safe zone of p^*. According to the definition of the safe zone of top-k result $\mathcal{RS}$ (Equation 19), object p does not contribute to the safe zone of the top-k result. □

Having the verification set, the $\mathcal{VO}$ for the safe zone can be constructed by applying Algorithm 1 with the following modification:

1) The input parameters are: MIRtreeNode N, top-k result $\mathcal{RS}$, thresholds $\{\tau_j\}$.
2) Replace str_{rs} with the traversal string str_{sz} for the safe zone.
3) Replace the condition in line 7 with Equation 12, where the minimum border distance of an entry e in the MIR-tree is computed as $bord_{min}(p^*, e) = \|p^*\ e\|_{min} w(p^*)/(w(p^*) + w(e))$.

The $\mathcal{VO}$ for the top-k result and the safe zone can be constructed using a single traversal of the MIR-tree. The traversal strings str_{rs} and str_{sz} indicate the relevant objects and entries for the top-k result and the safe zone, respectively, while only one copy of each object and entry are stored in the $\mathcal{VO}$.

3.3.2 Verifying a Safe Zone

The client re-constructs the safe zone using the $\mathcal{VO}$ and authenticates it. After obtaining the top-k objects, it calculates the safe zone using the top-k objects and the remaining objects in the $\mathcal{VO}$. Next, the client re-computes the root spatial and word digests using the str_{sz} and the $\mathcal{VO}$ by applying Algorithm 2, and it verifies them against the decrypted root spatial and word signatures, making sure that the entries in the $\mathcal{VO}$ are from the original MIR-tree. Then, it verifies the correctness of the safe zone by checking whether all the other entries in the $\mathcal{VO}$ except the top-k objects and influence objects satisfy Equation 12.

3.3.3 Soundness and Completeness

In this section, we prove that our proposal can authenticate a safe zone.

Proof of Soundness.
Suppose that the safe zone constructed by the client is fake or modified, i.e., the influence objects derived from the $\mathcal{VO}$ are fake or modified. The re-constructed root hash values cannot be verified against the root signatures provided by the data owner, which is detected, since the hash function is one way collision-resistant and the influence objects and their word weights must be used by Algorithm 2.

Proof of Completeness.
Let p be one of the influence objects and let N be the leaf node containing p. If all the entries in N are in the $\mathcal{VO}$, for sure p is in the $\mathcal{VO}$ and will be used to construct the safe zone. If N is pruned by the server, the MBR, keyword weights and digests of N are in the $\mathcal{VO}$. The client obtains the safe zone from the $\mathcal{VO}$, without using p. Since the re-constructed root hash values match the root signatures, the dominant regions of the top-k result object over N is correct and must intersect the safe zone, which informs the client about a potential violations of completeness.

3.4 MIR*-Tree

The $\mathcal{VO}$ size is critical to minimize the client/server communication cost and reduce the client-side result verification time. We proceed to propose an enhanced ADS, the MIR*-tree, that enables a reduction of the $\mathcal{VO}$ size. Specifically, the digests in the MIR*-tree are computed differently than in the MIR-tree. The underlying idea is applicable to any tree structure based ADS, e.g., the MR-tree [42], where each node contains multiple entries.

In the MIR-tree, each non-leaf entry has one spatial digest and several word digests. When a non-leaf entry (that is pruned) is added to the $\mathcal{VO}$, its MBR, one spatial digest, and m word digests are included, where m is the number of query keywords. Hence, for a visited non-leaf node in the MIR-tree, $n(m + 1)$ digests may be added to the $\mathcal{VO}$, where n is the number of pruned entries. Let S_d be the size in bytes of a digest and let S_m be the size in bytes of an MBR. The contribution of a non-leaf node N to the $\mathcal{VO}$ is $\mathcal{S}_N = n[(m + 1)S_d + S_m]$. With a node fanout in the range of hundreds, n can be large. And the size of a digest S_d is large (e.g., 64 bytes when using the SHA-512 hash function), while the size of an MBR S_m is at most 32 bytes if using double precision. Hence, the $\mathcal{VO}$ size is dominated by the number of digests included.

We propose a new way to compute the digests by incorporating the Rabin cryptosystem (quadratic residue) [31] to reduce the digests being added to the $\mathcal{VO}$, resulting in a new index structure, the MIR*-tree, where for each visited non-leaf node, at most $m + 1$ digests are added to the $\mathcal{VO}$. Then the contribution of a non-leaf node N in the MIR*-tree to the $\mathcal{VO}$ is $\mathcal{S}^*_N = (m + 1)S_d + n \cdot S_m$. Given an n value larger than 100, the $\mathcal{VO}$ size of the MIR*-tree can be significantly reduced, compared with that of the MIR-tree.

Finding a solution to $x^2 \equiv y \pmod{\mathbb{N}}$, where $\mathbb{N} = pq$ and p, q are primes, is equivalent to factoring a large number [31]. Thus, finding x under passive attacks is as hard as factoring. We proceed to exemplify how to compute the digests in the MIR*-tree using the quadratic residue. If an entry points to a leaf node, its digests are the same as those in the MIR-tree. If an entry points to a non-leaf node, its spatial digest is computed from the hash value of the concatenation of the binary representation of all the MBRs in its child node and one digest that is the modulo multiplication of all the digests' squares in its child node. As shown in Figure 8, the child node of R_5 contains four entries. The spatial digest of R_5 is computed as $R_5.H = h(R_1.\Lambda|R_2.\Lambda|R_3.\Lambda|R_4.\Lambda|H_s)$, where $H_s = \prod_{i=1}^{4} R_i.H^2 \pmod{\mathbb{N}}$. The word digests are computed in a similar way. Suppose all the five entries in Figure 8 contain word 'c' with weight 1. The word digest of R_5 for word 'c' is $h_c(R_5) = h((R_1, 1)|(R_2, 1)|(R_3, 1)|(R_4, 1)|H_w)$, where $H_w = \prod_{i=1}^{4} h_c(R_i)^2 \pmod{\mathbb{N}}$.

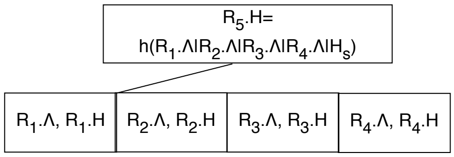
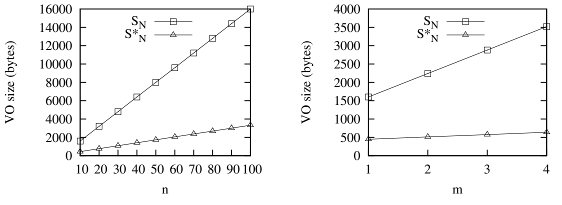


Fig. 8. Spatial Digests in the MIR*-Tree

During the process of constructing a $\mathcal{VO}$, when visiting a non-leaf node, the modulo multiplication of the spatial digests of pruned entries and the modulo multiplication of the word digests of pruned entries for each query keyword are added to the $\mathcal{VO}$. Consider the example in Figure 8, where all the five entries contain word 'c'. If entries R_1, R_2, and R_4 are pruned, we add *one* spatial digest $H'_s = \prod\{R_1.H, R_2.H,\ R_4.H\} \pmod{\mathbb{N}}$ and *one* word digest $H'_w = \prod\{h_c(R_1),\ h_c(R_2), h_c(R_4)\} \pmod{\mathbb{N}}$ to the $\mathcal{VO}$. In contrast, in the MIR-tree, three spatial digests, $R_1.H, R_2.H$, and $R_4.H$, and three word digests, $h_c(R_1), h_c(R_2)$, and $h_c(R_4)$, are added to the $\mathcal{VO}$.

The client is able to compute the spatial digest and the word digests for a node in a similar way as does the server using the $\mathcal{VO}$. Continuing with the above example, the client first re-computes the spatial digest and the word digest of R_4, i.e., $R_4.H$ and $h_c(R_4)$. Then $H_s = (H'_s \cdot R_4.H)^2 \pmod{\mathbb{N}}$ and $H_w = (H'_w \cdot h_c(R_4))^2 \pmod{\mathbb{N}}$. The correctness is guaranteed by $\prod h_i^2 \pmod{\mathbb{N}} = (\prod h_i)^2 \pmod{\mathbb{N}}$.

To observe the potential improvement, Figure 9(a) plots $\mathcal{S}_N$ and $\mathcal{S}^*_N$ as a function of n, where $S_m = 32$, $S_d = 64$, and $m = 1$. As n increases, i.e., more entries in a non-leaf node are pruned, the difference between $\mathcal{S}_N$ and $\mathcal{S}^*_N$ becomes larger. The size of the contribution of a non-leaf node in the MIR*-tree to the $\mathcal{VO}$ ($\mathcal{S}^*_N$) increases more slowly than that in the MIR-tree ($\mathcal{S}_N$). Figure 9(b) illustrates that $\mathcal{S}_N$ and $\mathcal{S}^*_N$ exhibit similar trends as the number of query keywords m increases. We expect that the MIR*-tree produces a smaller $\mathcal{VO}$ than does the MIR-tree, and the empirical study confirms our expectation.



(a) Varying n and $m = 1$ (b) Varying m and $n = 10$

Fig. 9. $\mathcal{S}_N$ vs. $\mathcal{S}^*_N$, $S_m = 32$, $S_d = 64$

4 Experimental Study

We conduct empirical studies to evaluate our proposals, comparing with two baselines introduced in Section 4.1. Section 4.2 presents the data sets, queries, parameters, and platform used in experiments. The proposals for the authentication of MkSK queries are evaluated in Section 4.3.

4.1 Baselines

We develop two baseline algorithms that exploit existing techniques for the authentication of MkSK queries. One is Inverted Index and Sorted Lists (IISL) utilizing signature chaining [26], and the other is Inverted Index and MR-tree (IIMR) that combines existing proposals [27], [42].

Inverted Index and Sorted Lists (IISL).
The IISL consists of two parts. One part is the inverted index [46], which is an efficient and popular index for keyword search, indexing the text descriptions of objects. The other part contains two sorted lists, indexing the spatial point locations of the objects. One list is for x-coordinates and the other one is for y-coordinates. Both lists are sorted ascendingly. We apply the idea of signature chaining to authenticate MkSK queries. A signature chain is constructed for each of the two sorted lists and each posting list in the inverted index.

Given an MkSK query q, we first compute its top-k result $\mathcal{RS}$ and the safe zone $\Upsilon^k(\mathcal{RS})$. In order to utilize the signature chaining to authenticate q, verification sets VS_X and VS_Y for the two sorted lists of coordinates and VS_{w_i} for the posting list of each query keyword in the inverted index are constructed to compute a $\mathcal{VO}$. We show how to compose the above verification sets in the appendix.

A $\mathcal{VO}$ is constructed by retrieving the objects covered by the verification sets. Each entry in the $\mathcal{VO}$ takes the form $(id, \Lambda,\ \{(w, weight)\})$, where id is the identifier, Λ is the minimum bounding rectangle, and $(w, weight)$ is a pair of a query keyword and its weight. In addition, a condensed RSA signature is computed by multiplying all the signatures of the objects in the $\mathcal{VO}$. The client recomputes the top-k result and the safe zone from the $\mathcal{VO}$ and verifies the condensed RSA signature.

Inverted Index and MR-tree (IIMR).
The IIMR consists of an inverted index that indexes the text descriptions of the objects and an MR-tree that indexes the spatial point locations of the objects. Verification sets VS_T and VS_s, presented in the appendix are used to construct a $\mathcal{VO}$ to authenticate an MkSK query result. Specifically, the efficient approach TNRA-Chain-MHT proposed by Pang et al. [27] is applied to the inverted index to authenticate the text relevancies of objects using VS_T, and the MR-tree authenticates the spatial distances of objects using VS_S.

4.2 Experiment Setup

We use two data sets, each containing objects with a point location and a text description, for studying the robustness and performance of the proposals. Data set EURO contains points of interest (e.g., ATMs, hotels, stores) in Europe[1]. Data set LONDFLI contains objects referring to photos taken in the area of London, where each object is composed by a spatial location and a text describing the photo (title and description). Table 3 offers additional details. We normalize object locations to fit a square domain with side length 10,000 meters.

[1] http://www.pocketgpsworld.com

TABLE 3
Data Set Statistics

data set	# of objects	# of distinct words	average # of words per object
EURO	162,033	35,315	18
LONDFLI	1,255,149	222,613	8

The Brinkhoff generator [5] is used to generate a trajectory using one location acquisition *per timestamp* (second). Each experiment has 100 moving queries with such trajectories. The keyword set of each query is generated by randomly picking an object in the data set and randomly choosing adjacent words from the document of the object.

We generally report the average value *per query per timestamp* of the following: (i) server-side elapsed time (for the query processing and the $\mathcal{VO}$ construction), (ii) server-side simulated I/O cost, (iii) client-side elapsed time (for the result verification), and (iv) communication cost (the $\mathcal{VO}$ size in KBytes).

We study the two baselines and our proposed solutions—the MIR-tree and the MIR*-tree. By default, we set the number k of results to 1, the number of query keywords to 2, the client moving speed to 10 m/s, and the fanout of the MIR-tree and the MIR*-tree to 200. We used disk-based index structures with a page size fixed at 4 KBytes. A simulated LRU buffer with size 256MB is used. All the algorithms are implemented in Java and run on a Linux machine with one processor (Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66 GHz) and 4 GB memory. The Java Virtual Machine Heap is set to 2 GB. We employ the SHA-512 as the hash function and 1024-bit RSA as the signature scheme.

4.3 Performance Results

We study the proposed methods under varying settings, including the performance on different data sets, varying numbers of requested results k, varying numbers of query keywords, varying speeds of moving queries, varying fanouts of the MIR-tree and the MIR*-tree, and varying data set sizes.

Evaluation on Different Data Sets.
Table 4 reports the average server elapsed time, server I/O cost, client elapsed time, number of entries in the $\mathcal{VO}$, $\mathcal{VO}$ size, index size, and communication frequency for the four methods on the two data sets. As expected, the server elapsed time is proportional to the server I/O cost. And the client elapsed time is correlated with the $\mathcal{VO}$ size. The MIR variants, including the MIR-tree and the MIR*-tree, outperform baseline IISL by orders of magnitude in all aspects. The reason why IISL has extremely poor performance is that its text and spatial score intervals are derived separately. Furthermore, its spatial score interval is converted into coordinate intervals on the x and y axes. Those intervals cover too many objects that are added to the $\mathcal{VO}$. Hence, its processing costs at both the server and client are high, and the $\mathcal{VO}$ size is large.

The server elapsed time and I/O cost of baseline IIMR are comparable to those the MIR variants, while the MIR variants outperform the IIMR by orders of magnitude in terms of the client elapsed time, number of entries in the $\mathcal{VO}$, and the $\mathcal{VO}$ size. Although the IIMR also derives the text score interval and the spatial score interval separately, its processing cost is reduced due to the use of the MR-tree. However, the separate intervals still cover too many objects that are added to the $\mathcal{VO}$, resulting in high verification cost at the client and large $\mathcal{VO}$ size.

The performance of the MIR-tree and the MIR*-tree are comparable in terms of server elapsed time and I/O cost. The numbers of entries in the $\mathcal{VO}$ for the MIR-tree and the MIR*-tree are the same. This is because the two indexes have similar structures and processing algorithms. The MIR*-tree differs from the MIR-tree in how digests are computed. In the MIR*-tree, fewer digests are added to the $\mathcal{VO}$ so that its client elapsed time and the $\mathcal{VO}$ size are significantly smaller than that for the MIR-tree.

In terms of index size, the IIMR is the smallest. The MIR-tree and the MIR*-tree occupy more space than does the IIMR due to the inverted files attached to non-leaf nodes. The IISL occupies substantial space due to the signature chains on the sorted lists of the x and y coordinates. These indexes, except for the IIMR, cannot fit in main memory.

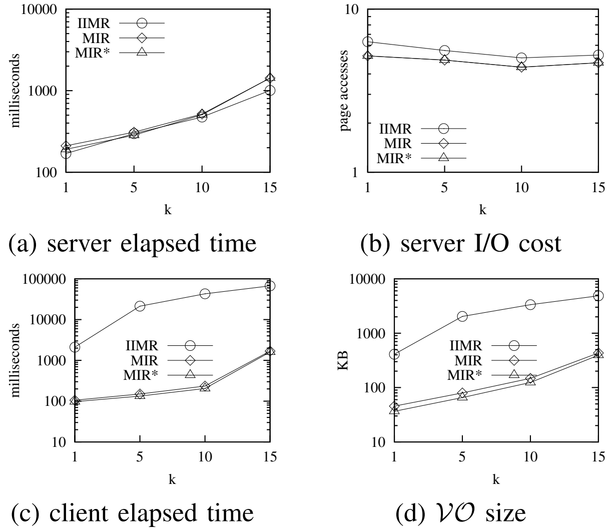
Our proposals outperform the two baseline algorithms consistently on the two data sets. We use LONDFLI as the default data set for the remaining experiments. Baseline IISL is omitted due to its poor performance and high storage cost.

TABLE 4
Performance Per Timestamp on Data Sets

Data Set		EURO	LONDFLI
Server elapsed time (milliseconds)	IISL	215.73	48966.67
	IIMR	20.03	169.69
	MIR	20.88	210.53
	MIR*	14.09	191.03
Server I/O cost (page accesses)	IISL	5611.06	2569.29
	IIMR	1.80	6.30
	MIR	1.63	5.17
	MIR*	1.63	5.17
Client elapsed time (milliseconds)	IISL	33.64	584.89
	IIMR	211.44	2104.79
	MIR	14.26	105.52
	MIR*	12.31	97.26
# of entries in the $\mathcal{VO}$	IISL	4195.86	30127.04
	IIMR	2965.34	8500.97
	MIR	293.08	756.31
	MIR*	293.08	756.31
$\mathcal{VO}$ size (KB)	IISL	173.13	1266.15
	IIMR	138.10	410.17
	MIR	15.89	45.53
	MIR*	13.90	36.67
Index size (GB)	IISL	2.97	635.05
	IIMR	0.24	1.68
	MIR	3.13	23.67
	MIR*	3.13	23.67
Communication frequency	All	0.034	0.032

Varying k.
Figure 10 shows the performance of the three approaches when varying the number k of requested results on data set LONDFLI. The MIR variants substantially and consistently

outperform the baseline in terms of client elapsed time and $\mathcal{VO}$ size. As expected, the server elapsed time, client elapsed time, and $\mathcal{VO}$ size of the three methods increase as k increases, since requesting more result objects increases the computation cost at the server side and calls for a larger $\mathcal{VO}$, resulting in higher verification cost at the client side. The server I/O costs of the three methods decrease slightly as k increases. This is mainly because more result objects produce a smaller safe zone, which enables more pruning. Hence, fewer index nodes (pages) are loaded from disk. However, the processing cost of each visited index node increases due to the larger result set.



(a) server elapsed time (b) server I/O cost

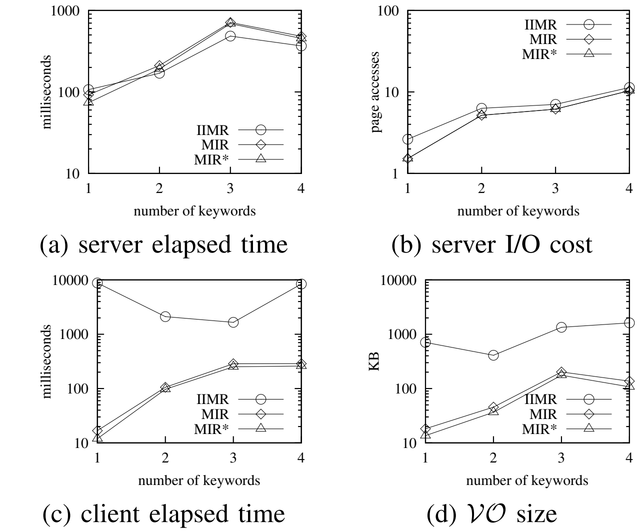
(c) client elapsed time (d) $\mathcal{VO}$ size

Fig. 10. Performance per Timestamp, Varying k

Varying the Number of Query Keywords.
Figure 11 shows the server elapsed time, server I/O cost, client elapsed time, and $\mathcal{VO}$ size of the three methods when varying the number of query keywords on data set LONDFLI. The MIR variants again consistently beat the baseline algorithm. As the number of query keywords increases, the performance of the three approaches decreases, since more objects are involved in the computations.

Varying the Speed of Moving Queries.
Figure 12 shows the performance of the three approaches when varying the speed of the moving queries on data set LONDFLI. We observe that as the speed increases, the computation and communication costs increase slightly, since a query with high speed more quickly leaves the safe zone and requests a query re-evaluation from the server. The MIR variants outperform the baseline significantly at different speeds.

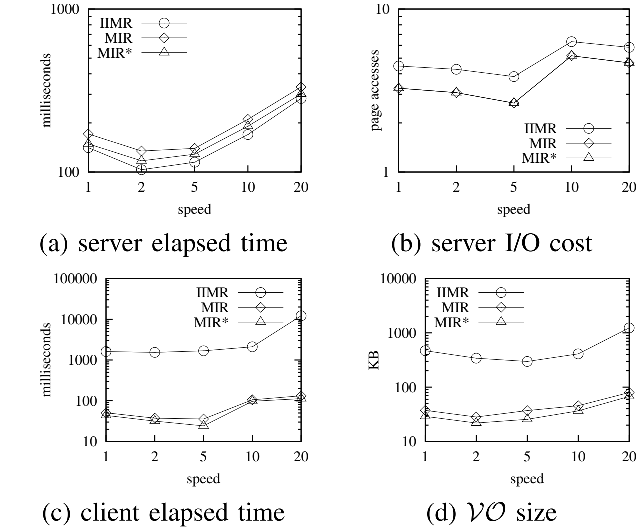
Varying the Fanout.
The ability of the MIR*-tree to reduce the $\mathcal{VO}$ size depends on the number of pruned entries in visited non-leaf nodes (Section 3.4). However, it is difficult to control the number of pruned entries, which may depend on several factors, e.g., the fanout, the query, and the distribution of the objects in the data set. This experiment studies how the performance of the MIR*-tree and the MIR-tree are affected by the fanout. Figure 13 shows the performance when varying the fanout on data set LONDFLI. The server elapsed time and the server I/O cost of the MIR-tree and the MIR*-tree are comparable, while the MIR*-tree has lower client elapsed time and smaller $\mathcal{VO}$ size than the MIR-tree. The reason is that in the MIR*-tree, fewer digests are added to the $\mathcal{VO}$, thus reducing the processing cost at the client and the $\mathcal{VO}$ size. And the operation of deriving one digest from multiple digests does not add much computational cost in the MIR*-tree. As the fanout increases, the server elapsed time and I/O cost decrease, since the heights of the MIR-tree and the MIR*-tree decrease so that it is faster to reach the leaf level to obtain results. In contrast, the client elapsed time and $\mathcal{VO}$ size increase as the fanout increases. Since a node contains more entries, more entries may be added to the $\mathcal{VO}$, it results in higher processing cost at the client side. The $\mathcal{VO}$ size of the MIR*-tree is 20% smaller than that of the MIR-tree when the fanout is 200. The improvement of the MIR*-tree decreases as the fanout increases. That is mainly because the number of pruned entries decreases.



(a) server elapsed time (b) server I/O cost

(c) client elapsed time (d) $\mathcal{VO}$ size

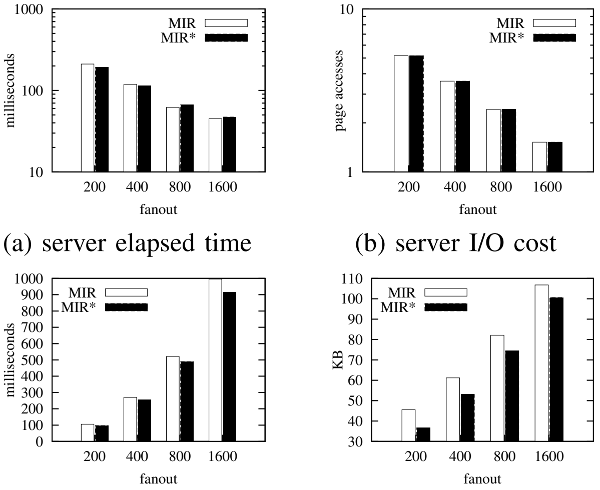
Fig. 11. Performance per Timestamp, Varying the Number of Keywords



(a) server elapsed time (b) server I/O cost

(c) client elapsed time (d) $\mathcal{VO}$ size

Fig. 12. Performance per Timestamp, Varying the Speed of Moving Queries

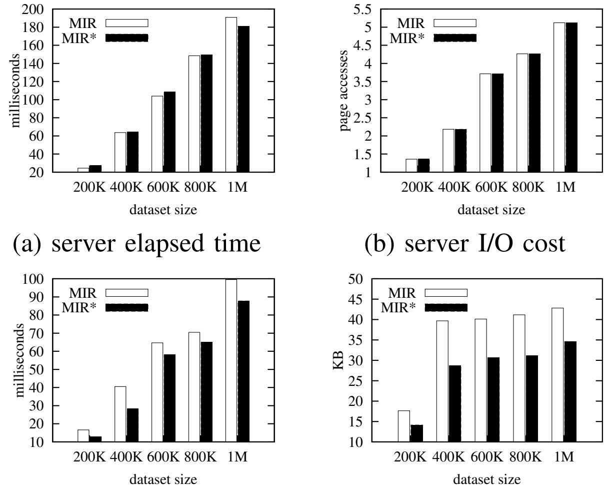


(a) server elapsed time (b) server I/O cost

(c) client elapsed time (d) $\mathcal{VO}$ size

Fig. 13. Performance per Timestamp, Varying Fanout

Scalability.
To observed the scalability of the proposed indexes, we generated 5 data sets from LONDFLI by randomly selecting 200K, 400K, 600K, 800K, and 1M objects. Figure 14 shows that the server elapsed time, server I/O cost, client elapsed time, and $\mathcal{VO}$ size exhibit increasing trends as with the data set size. Table 5 shows that the index size increases linearly as the data set size increases. All the indexes cannot fit the main memory.



(a) server elapsed time (b) server I/O cost

(c) client elapsed time (d) $\mathcal{VO}$ size

Fig. 14. Performance per Timestamp, Varying Data Set Size

TABLE 5
Index Size (GB)

	200K	400K	600K	800K	1M
MIR	4.03	7.91	12.13	15.76	19.71
MIR*	4.03	7.91	12.13	15.76	19.71

5 Related Work

5.1 Static Query Authentication

In general, existing query authentication techniques are classified into MHT-based approaches and approaches using signature chaining.

MHT-based Approaches. A popular authentication approach based on the MHT [23] has been used widely in many proposals [19], [27], [42], [43]. It is a hierarchical structure in which digests of nodes are recursively computed using a secure hash function from the leaf level to the root level. Then, the DO signs the root digest using a private key, e.g., RSA [33].

Devanbu et al. [9] authenticate one-dimensional range queries using an MHT on the query attribute. To prove the completeness of the result for a range query $[a, b]$ over a sorted list $(r_1, r_2, \ldots, r_n)$, the SP discloses the boundary records to the client. In other words, the query result is expanded into $(r_{i-1}, r_i, \ldots, r_j, r_{j+1})$, where $r_{i-1} < a \leq r_i$, $r_j \leq b < r_{j+1}$, and $1 < i \leq j < n$. The boundary records guarantee that no records in the query range are omitted. In addition to the expanded result, for each node visited in the MHT, digests of all left-siblings of the left boundary record r_{i-1} and digests of all right-siblings of the right boundary record r_{j+1} on the paths from the root are added to the $\mathcal{VO}$. If the re-constructed root digest from the $\mathcal{VO}$ matches the decrypted root signature, the result is sound. Later, two disk-based adaptations of the MHT, i.e., the VB-tree [28] and the Merkle B-tree (MB-tree) [19], has been are proposed.

Pang et al. [27] apply the MHT to the inverted index and documents to authenticate text retrieval query results (keyword search). The proposed Term-MHT builds an MHT on the posting list of each word, while the Term-CMHT constructs an MHT on the posting list entries in each disk page, since a posting list normally occupies several disk pages. The digest of the MHT (hash value of the root) for a disk page is integrated into the MHT for the previous disk page. The digest of the first page of each posting list is signed by the DO, which is used to verify any j leading pages of a posting list. A document is represented by a sequence of entries $(w,\ weight)$ in ascending w order. The proposed Document-MHT takes the document entries as leaves. Both the Term-MHT and Term-CMHT are used to guarantee the that posting lists involved in the query processing are correct, and the Document-MHT is used to check the computed score of each encountered document.

Another solution inspired by the MHT technique is the MR-tree [42], an index based on the R*-tree that is capable of authenticating arbitrary spatial queries. Its leaf nodes are identical to those of the R*-tree. Non-leaf nodes contain entries of the form $e = (ptr, \Lambda, H)$, where $e.ptr$ is a reference to a child node, $e.\Lambda$ is the MBR of all rectangles in entries of the child node, and $e.H$ is a digest of the child node. The digest of a leaf node is computed from the hash value of the concatenation of the binary representation of all objects in the node. The digest of a non-leaf node summarizes its child nodes' MBRs as well as their digests. The digest of the root node is signed by the DO and is stored with the tree.

For the client to verify a query result, the SP utilizes the MR-tree to generate a verification object $\mathcal{VO}$. A circular verification set $\odot(q, \gamma)$ with center q and radius equal to the distance between q and its k^{th} nearest neighbor is defined.

The MR-tree is traversed in depth-first manner with the following conditions: (1) a non-leaf entry e is added to the $\mathcal{VO}$ if e does not intersect $\odot(q, \gamma)$; (2) all objects in a visited leaf node are added to the $\mathcal{VO}$. To authenticate the query result, the client first reconstructs the digest of the root of the MR-tree from $\mathcal{VO}$. Then it verifies it against the root signature offered by the data owner. Then the client finds the kNNs directly from the $\mathcal{VO}$, defines its own verification set $\odot(q, \gamma')$, and checks whether no non-leaf entries in $\mathcal{VO}$ intersects $\odot(q, \gamma')$. If yes, the correctness of the NN result is guaranteed.

Lin et al. [22] study the authentication of location-based skyline queries. They propose two authentication methods. One is based on the MR-tree. The other one utilizes the MR-Sky-tree that is designed specially for skyline queries.

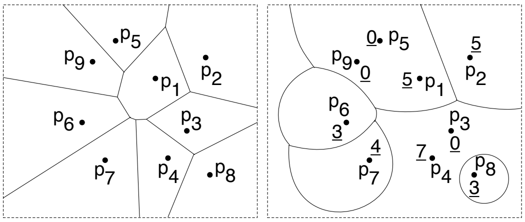
Signature Chaining. Unlike MHT, signature chaining [26] authenticates both the soundness and completeness of one-dimensional range queries. Consider a data set containing 4 records p_1–p_4, sorted on the query attribute. The data owner inserts two special records p_0 and p_5 with values $-\infty$ and $+\infty$. A signature is created for each record, except the special records. The signature of a record p_i is based on the hash value of the concatenation of its preceding record p_{i-1}, itself p_i, and its succeeding record p_{i+1}. The data set and the signatures are sent to the SP. Assume that the result of a range query contains p_2, p_3, and p_4. The SP returns a $\mathcal{VO}$ containing the result (p_2, p_3, p_4), the signature of each record in the result $(sig_{123}, sig_{234}, sig_{345})$, and the boundary records p_1 and p_5. In order to reduce the $\mathcal{VO}$ size, the Condensed-RSA [24] adds only one signature $sig = sig_{123} \cdot sig_{234} \cdot sig_{345} \pmod{n}$ to the $\mathcal{VO}$ instead of three. Given a $\mathcal{VO}$, the client first checks that the signature is valid, which guarantees that all the records in the $\mathcal{VO}$ are correct and adjacent. Then the client checks that the two boundary records fall outside the query range, which ensures that no result record is missing at the range boundaries, i.e., p_2 and p_4 are indeed the first and last records of the result.

This scheme has also been extended to multi-dimensional index structures [7]. Compared with the MHT-based approaches, signature chaining methods incur high index construction costs, storage overhead, and client-side verification time [19].

The two baseline algorithms considered in the experimental study exploit the above authentication techniques (MHT, MR-tree, and signature chaining), and they are found to be inefficient compared to our proposals.

5.2 Moving Query Authentication

Existing technique for authenticating moving kNN queries [44] cannot help in authenticating MkSK queries, since the safe zone of a kNN query is a Voronoi cell [25], while the safe zone of an MkSK query is a Multiplicatively Weighted Voronoi (MW-Voronoi) cell [25]. A Voronoi cell considers only spatial distance and has a polygon shape. In contrast, an MW-Voronoi cell defines the influence region of an object based on its weight (e.g., text relevance) and is an irregular shape consisting of edges, arcs and holes. The structures and construction algorithms for these different kinds of cells are totally different. Figure 15(b) illustrates the MW-Voronoi diagram of the same nine objects as in Figure 15(a). In the MW-Voronoi diagram, each object has a weight (underlined numbers). These weights are the text relevancies between the objects and the query, known only when the query is submitted. In other words, different queries produce different MW-Voronoi diagrams on the same data set. Thus, pre-computation based approaches [17], [44] are inapplicable, since enumerating all possible MW-Voronoi diagrams is computationally prohibitive.



(a) Voronoi Diagram (b) MW-Voronoi Diagram

Fig. 15. Voronoi Versus MW-Voronoi Diagrams

Yung et al. [45] study the authentication of moving range queries. The safe zone of a moving range query and its verification set are also different from ours, and thus their technique is inapplicable to our problem.

6 Conclusion

This paper proposes a new authenticated data structure, the MIR-tree, to efficiently authenticate moving top-k spatial keyword query results, thus guaranteeing the soundness and completeness of both a top-k result and its corresponding safe zone. A verification object for authenticating the top-k results and safe zones of MkSK queries is designed. Algorithms for constructing and using verification objects for verifying the top-k results and safe zones are developed. An enhancement of the MIR-tree, the MIR*-tree, is proposed to further reduce the communication cost. Extensive empirical studies on real data sets demonstrate that the proposed approaches are capable of outperforming two baseline algorithms that utilize existing techniques by orders of magnitude.

References

[1] Proposed federal information processing standard for digital signature standard (dss). *Federal Register*, 56(169):42980–42982, 1991.

[2] Secure hashing algorithm. *National Institute of Science and Technology*, FIPS 180-2, 2001.

[3] F. Aurenhammer and H. Edelsbrunner. An optimal algorithm for constructing the weighted Voronoi diagram in the plane. *Pattern Recognition*, 17(2): 51–57, 1984.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432, 2003.

[5] T. Brinkhoff. A framework for generating network-based moving objects. *Geoinformatica*, 6(2):153–180, 2002.

[6] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD*, pages 277–288, 2006.

[7] W. Cheng and K.-L. Tan. Query assurance verification for outsourced multi-dimensional databases. *Journal of Computer Security*, 17(1):101–126, 2009.

[8] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[9] P. T. Devanbu, M. Gertz, C. U. Martel, and S. G. Stubblebine. Authentic data publication over the internet. *Journal of Computer Security*, 11(3):291–314, 2003.

[10] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[11] Z. Fan, Y. Peng, B. Choi, J. Xu, and S. S. Bhowmick. Towards efficient authenticated subgraph query service in outsourced graph databases. *IEEE Transactions on Services Computing*, to appear, 2013.

[12] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[13] R. Gambini, D. L. Huff, and G. F. Jenks. Geometric properties of market areas. *Regional Science*, 20(1):5–92, 1968.

[14] Google. Google Places. *Stats & Facts*, 2012.

[15] H. Hacigümüs, S. Mehrotra, and B. R. Iyer. Providing database as a service. In *ICDE*, pages 29–38, 2002.

[16] H. Hu, J. Xu, Q. Chen, and Z. Yang. Authenticating location-based services without compromising location privacy. In *SIGMOD*, pages 301–312, 2012.

[17] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi. Verifying spatial queries using Voronoi neighbors. In *GIS*, pages 350–359, 2010.

[18] D. L. Huff. The delineation of a national system of planning regions on the basis of urban spheres of influence. *Regional Science*, 7(3): 323–329, 1973.

[19] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. In *SIGMOD*, pages 121–132, 2006.

[20] F. Li, K. Yi, M. Hadjieleftheriou, and G. Kollios. Proof-infused streams: Enabling authentication of sliding window queries on streams. In *VLDB*, pages 147–158, 2007.

[21] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. L. Lee, and X. Wang. IR-tree: An efficient index for geographic document search. *IEEE Trans. Knowl. Data Eng.*, 23(4):585–599, 2011.

[22] X. Lin, J. Xu, and H. Hu. Authentication of location-based skyline queries. In *CIKM*, pages 1583–1588, 2011.

[23] R. C. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.

[24] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. *Trans. Storage*, 2:107–138, 2006.

[25] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, 2nd edition, 2000.

[26] H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan. Verifying completeness of relational query results in data publishing. In *SIGMOD*, pages 407–418, 2005.

[27] H. Pang and K. Mouratidis. Authenticating the query results of text search engines. *PVLDB*, 1(1):126–137, 2008.

[28] H. Pang and K.-L. Tan. Authenticating query results in edge computing. In *ICDE*, pages 560–571, 2004.

[29] S. Papadopoulos, Y. Yang, and D. Papadias. Cads: Continuous authentication on data streams. In *VLDB*, pages 135–146, 2007.

[30] Y. Peng, Z. Fan, B. Choi, J. Xu, and S. S. Bhowmick. Authenticated subgraph similarity search in outsourced graph databases. *IEEE Trans. Knowl. Data Eng.*, to appear, 2014

[31] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Cambridge, MA, USA, 1979.

[32] R. Rivest. Rfc 1321: The md5 message-digest algorithm. *Internet Activities Board*, 1992.

[33] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[34] Search Engine Land. Microsoft: 53 percent of mobile searches have local intent, 2012.

[35] Z. Song and N. Roussopoulos. K-nearest neighbor search for moving query point. In *SSTD*, pages 79–96, 2001.

[36] L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Commun.*, 29(18):3812–3824, 2006.

[37] D. Wu, G. Cong, and C. S. Jensen. A framework for efficient spatial web object retrieval. *VLDB J.*, 21(6):797–822, 2012.

[38] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. Joint top-k spatial keyword query processing. *IEEE Trans. Knowl. Data Eng.*, 24(10):1889–1903, 2012.

[39] D. Wu, M. L. Yiu, C. S. Jensen. Moving spatial keyword queries: formulation, methods, and analysis. *TODS*, 38(1):7, 2013

[40] D. Wu, M. L. Yiu, C. S. Jensen, and G. Cong. Efficient continuously moving top-k spatial keyword query processing. In *ICDE*, pages 541–552, 2011.

[41] Y. Yang, D. Papadias, S. Papadopoulos, and P. Kalnis. Authenticated join processing in outsourced databases. In *SIGMOD*, pages 5–18, 2009.

[42] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios. Authenticated indexing for outsourced spatial databases. *VLDB J.*, 18(3):631–648, 2009.

[43] M. L. Yiu, Y. Lin, and K. Mouratidis. Efficient verification of shortest path search via authenticated hints. In *ICDE*, pages 237–248, 2010.

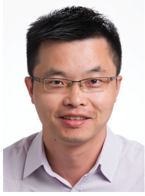
[44] M. L. Yiu, E. Lo, and D. Yung. Authentication of moving knn queries. In *ICDE*, pages 565–576, 2011.

[45] D. Yung, E. Lo, and M. L. Yiu. Authentication of moving range queries. In *CIKM*, pages 1372–1381, 2012.

[46] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), 2006.

Dingming Wu is a Post-doc Fellow of Computer Science at The University of Hong Kong. She was a Post-doc Teaching Fellow at Hong Kong Baptist University for a 2-year period from 2011 to 2013. She received the PhD degree in computer science from Aalborg University, Denmark in 2011, under the supervision of Prof. Christian S. Jensen. She received the Bachelor degree and the Master degree in computer science from Huazhong University of Science and Technology and Peking University in 2005 and 2008, respectively. Her research concerns spatial keyword search, geo-social networks, and recommendation systems.

Byron Choi received the bachelor of engineering degree in computer engineering from the Hong Kong University of Science and Technology (HKUST) in 1999 and the MSE and PhD degrees in computer and information science from the University of Pennsylvania in 2002 and 2006, respectively. He is now an associate professor in the Department of Computer Science at the Hong Kong Baptist University.



Jianliang Xu is a professor in the Department of Computer Science, Hong Kong Baptist University. His research interests include data management, mobile/pervasive computing, and networked and distributed systems. He has published more than 120 technical papers in these areas. He is currently serving on the editorial boards of IEEE Transactions on Knowledge and Data Engineering and International Journal of Distributed Sensor Networks. He is a senior member of IEEE.

Christian S. Jensen is an Obel Professor of Computer Science at Aalborg University, Denmark. He was a Professor at Aarhus University for a 3-year period from 2010 to 2013, and he was previously at Aalborg University for two decades. He recently spent a 1-year sabbatical at Google Inc., Mountain View. His research concerns data management and data-intensive systems, and its focus is on temporal and spatio-temporal data management. Christian is an ACM and an IEEE fellow, and he is a member of the Academia Europaea, the Royal Danish Academy of Sciences and Letters, and the Danish Academy of Technical Sciences. He has received several national and international awards for his research. He is Editor-in-Chief of ACM TODS and an Editor-in-Chief of The VLDB Journal.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2350252, IEEE Transactions on Knowledge and Data Engineering

APPENDIX A
SAFE ZONE

A safe zone or the query considered is an order-k multiplicatively weighted Voronoi (MW-Voronoi) region [25]. Here, we recall relevant definitions.

Let $\mathcal{D}$ be a set of weighted points in two-dimensional Euclidean space $\mathcal{U}$. A point a in $\mathcal{D}$ has (i) a weight $w(a)$ and (ii) coordinates (a_x, a_y). The weighted distance between any point z in $\mathcal{U}$ and a is defined as $d_w(z,a) = \|z\ a\|/w(a)$, where $\|z\ a\|$ is the Euclidean distance between z and a.

Definition A.1: The *dominant region* of point a over point b is defined as:

$$Dom_{a,b} = \{z \in \mathcal{U} \mid d_w(z,a) \leq d_w(z,b)\}. \tag{13}$$

To characterize the shape of $Dom_{a,b}$, we define the Apollonius circle and explain its relationship with $Dom_{a,b}$. As an example, Figure 16 shows the Apollonius circle of two points a and b. Any location z on the circle satisfies the equation $\|z\ a\| = \mu \cdot \|z\ b\|$.

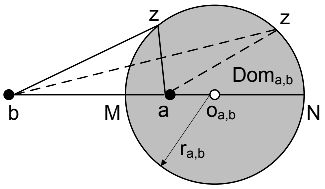


Fig. 16. Apollonius circle

Definition A.2: Given two points a and b and a constant $0 \leq \mu \leq 1$, the *Apollonius circle* $C_{a,b}$ is defined as all locations z that satisfy: $\|z\ a\| = \mu \cdot \|z\ b\|$. The *Apollonius circular region* $C_{a,b}$ is defined as the region where $\|z\ a\| \leq \mu \cdot \|z\ b\|$.

Lemma A.1: **[Radius and center, Apollonius circle]** Given two points a and b such that $w(a) < w(b)$, $Dom_{a,b} = C_{a,b}$. The center $o_{a,b}$ and the radius $r_{a,b}$ of $C_{a,b}$ are shown in Equations 14 and 15. In addition, we have: $Dom_{b,a} = \mathcal{U} - C_{a,b}$, where $\mathcal{U}$ is the spatial domain.

$$o_{a,b} = \left(\frac{w^2(b)\cdot a_x - w^2(a)\cdot b_x}{w^2(b)-w^2(a)}, \frac{w^2(b)\cdot a_y - w^2(a)\cdot b_y}{w^2(b)-w^2(a)}\right) \tag{14}$$

$$r_{a,b} = \frac{w(a)\cdot w(b)\cdot \|a\ b\|}{w^2(b)-w^2(a)} \tag{15}$$

We will use "circle" and "circular region" interchangeably.

For the special case $w(a) = w(b)$, the Apollonius circular region $C_{a,b}$ degenerates to the perpendicular half plane $\perp_{a,b}$. In general, the dominant region $Dom_{a,b}$ is expressed as:

$$Dom_{a,b} = \begin{cases} C_{a,b} & \text{if } w(a) < w(b) \\ \mathcal{U} - C_{b,a} & \text{if } w(a) > w(b) \\ \perp_{a,b} & \text{if } w(a) = w(b) \end{cases} \tag{16}$$

The MW-Voronoi diagram of $\mathcal{D}$ is the collection of MW-Voronoi regions of all points in $\mathcal{D}$. These regions form a (disjoint and complete) partitioning of the spatial domain.

Definition A.3: Given a (result) point $p^* \in \mathcal{D}$, its *MW-Voronoi region* with respect to $\mathcal{D}$ is defined as:

$$\Upsilon(p^*) = \bigcap_{p' \in \mathcal{D} - \{p^*\}} Dom_{p^*,p'} \tag{17}$$

Based on a (result) point $p^* \in \mathcal{D}$, we can partition $\mathcal{D}$ into $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^- \cup \mathcal{D}^o \cup \{p^*\}$, where set D^+ contains all points with higher weight than p^*, set D^- contains all points with lower weight than p^*, and set $\mathcal{D}^o$ contains the objects whose weights are identical to p^*. By applying Equation 16, we can re-express the MW-Voronoi region as Equation 18. Thus, higher-weight neighbors add to the MW-Voronoi region of a point, forming convex edges; lower-weight neighbors subtract from it, forming concave edges; and equal-weight neighbors crop it with straight lines.

$$\begin{aligned} \Upsilon(p^*) &= \bigcap_{p \in \mathcal{D} - \{p^*\}} Dom_{p^*,p} \\ &= \bigcap_{p_j \in \mathcal{D}^+} C_{p^*,p_j} \cap \bigcap_{p_i \in \mathcal{D}^o} \perp_{p^*,p_i} \cap \bigcap_{p_k \in \mathcal{D}^-} (\mathcal{U} - C_{p_k,p^*}) \\ &= \bigcap_{p_j \in D^+} C_{p^*,p_j} \cap \bigcap_{p_i \in \mathcal{D}^o} \perp_{p^*,p_i} - \bigcup_{p_k \in D^-} C_{p_k,p^*} \end{aligned} \tag{18}$$

In the example in Figure 17, the gray region is the safe zone (MW-Voronoi region) $\Upsilon(p_1)$ for the top-1 result p_1. It is calculated by $C_{p_1,p_4} \cap C_{p_1,p_5} \cap \perp_{p_1,p_2} - C_{p_3,p_1}$. It is generally far from every data object that contributes to defining a safe zone. Note that safe zone $\Upsilon(p_1)$ consists of 3 arcs (edges) that originate from the Apollonius circles C_{p_1,p_4} and C_{p_3,p_1} and the perpendicular bisector $\perp_{p_1,p_2}$. Apollonius circle C_{p_1,p_5} does not contribute to the safe zone $\Upsilon(p_1)$ because $\Upsilon(p_1)$ is completely inside C_{p_1,p_5}. The objects that are necessary for representing a safe zone are called influence objects. In Figure 17, objects p_2, p_3, and p_4 are influence objects, while p_5 is not.

Let $\mathcal{RS}$ be the top-k result of a query q. According to Okabe et al. [25], the order-k MW-Voronoi region of the set $\mathcal{RS}$, denoted by $\Upsilon^k(\mathcal{RS})$, contains all locations that take the set $\mathcal{RS}$ as the top-k result. In other words, $\Upsilon^k(\mathcal{RS})$ is the safe zone for the result set $\mathcal{RS}$. Region $\Upsilon^k(\mathcal{RS})$ can be expressed by the intersections of order-1 MW-Voronoi regions [25]

$$\Upsilon^k(\mathcal{RS}) = \bigcap_{p_j^* \in \mathcal{RS}} \Upsilon_{\mathcal{D}-\mathcal{RS}}(p_j^*), \tag{19}$$

where $\Upsilon_{\mathcal{D}-\mathcal{RS}}(p_j^*)$ denotes the MW-Voronoi region of p_j^* with respect to the object set $\mathcal{D} - \mathcal{RS}$.

APPENDIX B
VERIFICATION REGIONS FOR BASELINES

Given the top-k result $\mathcal{RS}$ of q, let T_k be a text score interval $[ts_k, +\infty)$ and S_k be a spatial score interval $[0, ss_k]$, where $ts_k = \min_{p \in \mathcal{RS}} \{tr_q(p)\}$ refers to the minimum text relevance of the objects in $\mathcal{RS}$ and ss_k refers to the

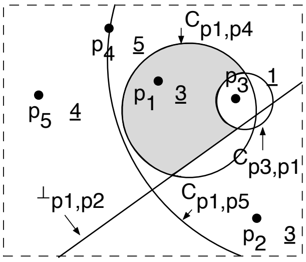


Fig. 17. Example Safe Zones

TABLE 6
Symbols

p	an object
ψ	keywords
λ	point location
$\mathcal{D}$	data set
$\mathcal{RS}$	top-k object set
$\Upsilon(\cdot)$	safe zone
$\|\cdot\ \cdot\|$	Euclidean distance
$\|\cdot\ \cdot\|_{min}$	minimum Euclidean distance
$tr_q(p)$	text relevancy of p w.r.t. q
$rank_q(p)$	ranking score of p w.r.t. q
$w(\cdot)$	weight
$Dom_{p^*,p}$	dominant region of p^* over p
$C_{p^*,p}$	Apollonius circular region
$\mathcal{U}$	Euclidean space
$\perp_{p^*,p}$	perpendicular half plane
$Q(\lambda,\psi,k)$	top-k result of a query with keywords ψ and point location λ
$h_w(\cdot)$	word digest
$h(\cdot)$	secure hash function
Λ	minimum bounding rectangle
$bord_{min}(p^*,p)$	minimum distance between p^* and the border of dominant region $Dom_{p^*,p}$
$bord_{max}(p^*,p)$	maximum distance between p^* and the border of dominant region $Dom_{p^*,p}$
VS_{rs}	verification set for top-k result objects
VS_{sz}	verification set for safe zone
VS_X	verification set for x-coordinate
VS_Y	verification set for y-coordinate
VS_w	verification set for keyword w
VS_T	verification set for text relevance
VS_S	verification set for spatial distance

maximum distance between the query and the result objects, i.e., $\max_{p\in\mathcal{RS}}\{\|q\ p\|\}$.

Given the safe zone of the top-k result $\Upsilon^k(\mathcal{RS})$, let I be the influence object set of the safe zone. We define two kinds of distances. One is the minimum border distance of an object p from the result set $\mathcal{RS}$, which is the minimum value of the minimum border distances of p from all the result objects in $\mathcal{RS}$, i.e., $bord_{min}(\mathcal{RS},p)=\min_{p_j^*\in\mathcal{RS}}\{bord_{min}(p_j^*,p)\}$. The other is the maximum distance between the result set $\mathcal{RS}$ and an object p, i.e., $\|\mathcal{RS}\ p\|_{max}$. Consider the non-influence object $p_s\in(\mathcal{D}\setminus\mathcal{RS}\setminus I)$ with the smallest minimum border distance from $\mathcal{RS}$, i.e., $\forall p\neq p_s\wedge p\in(\mathcal{D}\setminus\mathcal{RS}\setminus I), bord_{min}(\mathcal{RS},p_s)<bord_{min}(\mathcal{RS},p)$. We construct a text score interval $T_{sz}=(ts_{sz},+\infty)$ and a spatial score interval $S_{sz}=[0,ss_{sz})$, where $ts_{sz}=tr_q(p_s)$ is the text relevance of p_s and $ss_{sz}=\|\mathcal{RS}\ q\|_{max}+\|\mathcal{RS}\ p_s\|_{max}$. Theorem B.1 guarantees that any object with text relevance no greater than ts_{sz} and having the distance from q no smaller than ss_{sz} cannot affect the safe zone $\Upsilon^k(\mathcal{RS})$.

Theorem B.1: Consider the non-influence object $p_s\in(\mathcal{D}\setminus\mathcal{RS}\setminus I)$, such that $\forall p\neq p_s\wedge p\in(\mathcal{D}\setminus\mathcal{RS}\setminus I)$ and $bord_{min}(\mathcal{RS},p_s)<bord_{min}(\mathcal{RS},p)$. Let $ts_{sz}=tr_q(p_s)$ and $ss_{sz}=\|\mathcal{RS}\ q\|_{max}+\|\mathcal{RS}\ p_s\|_{max}$. Object p cannot affect the safe zone $\Upsilon^k(\mathcal{RS})$ if $tr_q(p)\le ts_{sz}$ and $\|q\ p\|\ge ss_{sz}$.

Proof: Since p_s is a non-influence object, it cannot affect the safe zone, i.e., $\bigvee_{p_j^*\in\mathcal{RS}} bord_{min}\ (p_j^*,p_s)>\tau_j$. Consider an object p such that $tr_q(p)\le ts_{sz}=tr_q(p_s)$ and $\|q\ p\|\ge ss_{sz}=\|\mathcal{RS}\ q\|_{max}+\|\mathcal{RS}\ p_s\|_{max}$. Applying the triangle inequality, for any result object p_j^* in $\mathcal{RS}$, we have $\|q\ p_j^*\|+\|p_j^*\ p\|>\|q\ p\|\ge\|\mathcal{RS}\ q\|_{max}+\|\mathcal{RS}\ p_s\|_{max}\ge\|q\ p_j^*\|+\|p_j^*\ p_s\|$. Thus, we derive $\|p_j^*\ p\|>\|p_j^*\ p_s\|$. Since $bord_{min}(p_j^*,p)=\|p_j^*\ p\|tr_q(p_j^*)/\ (tr_q(p_j^*)+tr_q(p))>\|p_j^*\ p_s\|tr_q(p_j^*)/(tr_q(p_j^*)+tr_q(p_s))=bord_{min}(p_j^*,p_s)>\tau_j$, object p cannot affect the safe zone. □

Combining the text score intervals and the spatial score intervals derived from the top-k result and the safe zone, we have verification sets

$$VS_T=T_k\cup T_{sz}=[ts,+\infty)$$

and

$$VS_S=S_k\cup S_{sz}=[0,ss],$$

where $ts=\min\{ts_k,ts_{sz}\}$ and $ss=\max\{ss_k,ss_{sz}\}$. The verification sets for the sorted lists of coordinates are the coordinate intervals

$$VS_X=[q_x-ss,q_x+ss]$$

and

$$VS_Y=[q_y-ss,q_y+ss],$$

which cover the x-coordinates and y-coordinates, respectively. The coordinates of q are represented as (q_x,q_y). As shown in Figure 18, the shaded circle is VS_S. The VS_X and VS_Y are the bold intervals on the x and y axes.

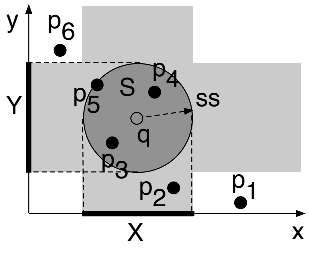


Fig. 18. Example Spatial Score Interval

To compose the verification set VS_{w_i} for the posting list of each query keyword w_i in the inverted index, the state-of-the-art algorithm Threshold with No Random Access (TNRA) [10] is applied on the inverted index to retrieve the objects whose text relevancies fall inside VS_T. When the TNRA algorithm stops, the posting list of each query keyword has a cut-off value cv_i that indicates all the objects whose text relevancies fall inside VS_T have been obtained, i.e., all the objects after the cut-off values are outside VS_T. Then the verification set on the posting list of word w_i is:

$$VS_{w_i}=[cv_i,+\infty).$$