Unified Index for Mobile Object Data and Authorizations*

Vijayalakshmi Atluri and Qi Guo

Rutgers University, Newark NJ 07012, USA
{atluri, qiguo}@cimic.rutgers.edu

Abstract. Often, enforcing security incurs overhead, and as a result may degrade the performance of a system. In this paper, we attempt to address this problem in the context of enforcing access control policies in a mobile data object environment. There are a number of applications that call for fine-grained specification of security policies in guaranteeing the confidentiality of data or privacy of individuals in a mobile environment. In particular, the security policies state the rules for providing controlled access to the mobile user profiles, to their current location and movement trajectories, to mobile resources, and stationary resources based on the mobile user location. Either a subject or an object in an authorization specification can be a moving object. The access requests in such an environment can typically be based on *past, present* and *future* status of the moving objects. To effectively serve such access requests, one must efficiently organize the mobile objects as well as authorizations.

Although implementation of authorizations as access control list, capability list or access matrix is suitable for traditional data, it is not suitable to search mobile object authorizations as they are based on spatial and temporal attributes of subjects and objects, rather than subject and object identifiers. When a subject issues an access request, the system must first retrieve the relevant objects from the moving object database, and then verify whether there exists an authorization that allows the subject to access these objects. Since both the moving objects and authorizations are spatiotemporal in nature, for efficient processing of access requests, it is essential that they both be organized using some index structures. As a result, processing an access request requires searching two indexes - one, the moving object index, and the other, the authorization index. To improve the response time of access requests, in this paper, we propose a unified index structure, called ^{S}TPR-tree to index both moving objects and authorizations that govern access to them. As a result of the unified index, access requests can be processed in one pass, thereby improving the response time. Note that current access control systems do not use any index for authorizations; our work is a step in this direction. We show how the ^{S}TPR-tree can be constructed and maintained, and provide algorithms to process access requests.

1 Introduction

Recent advances to mobile communication, Global Positioning System (GPS) and Radio Frequency Identification (RFID) technologies have propelled the growth of a number of mobile services. Among them, *location-based service* (LBS) is becoming the

* This work is supported in part by the National Science Foundation under grant IIS-0242415.

S. De Capitani di Vimercati et al. (Eds.): ESORICS 2005, LNCS 3679, pp. 80–97, 2005.

most widely used services. LBS presents a major new market for the mobile industry, which includes (i) navigation services, (ii) providing business descriptions in a given geographical radius, real-time alerts on traffic conditions and information about highway services, (iii) personalized point-of-need information delivery, such as travel reservations, new or interesting products and services, and (iv) mobile advertising, which includes personalized, location-aware, and context-sensitive advertising, based on mobile customer profiles and preferences, as mobile devices are ideal for marketing channels for impulse buying [19]. For example, a wireless shopping site can be designed to present users with targeted content such as clothing items on sale, based on prior knowledge of their preferences and/or knowledge of their current location, such as proximity to a shopping mall [25]. LBS can be also be used in emergency situations to transmit messages via a reverse 911 service to individuals. To deliver LBS, service providers require access to customers' preference profiles either through a proprietary database or through an arrangement with an LBS provider, who matches customer profiles to vendor offerings. In order to implement such services, customization and personalization based on the location information, customer needs, and vendor offerings are required. Industrial and corporate applications, including tracking of material through the supply chain and inventory, and tracking physicians, patients, and equipment in a hospital. This relies on the deployment of RFID technologies, which is becoming inexpensive and will likely be used by a number of retail businesses.

In all the above applications, the mobile objects may include mobile phones, wireless PDAs, GPS equipped units such as boats, trucks, automobiles, airplanes, and soldiers, objects with RFID tags, a variety of moving sensors and other wireless computing devices.

In delivering mobile services, one encounters a number of security and privacy concerns, which are discussed below.

- **Location privacy:** Privacy of mobile users can be compromised by disclosing the location and movement. Note that it is essential to identify the location of the mobile object due to the following two reasons. First, to effectively function, location-based services require information about the location of the communication device. Second, in countries like U.S., the European Union and Japan, laws require that mobile telephones be able to provide location data with a fairly detailed accuracy for the purposes of emergency situations.

  Although identifying (and sometimes tracking) of the location of a mobile object is essential in delivering a mobile service, it could pose a threat to privacy of the person carrying the mobile device.

  Unlike the internet, location information has the potential to allow an adversary to physically locate a person, and therefore wireless subscribers carrying mobile devices have legitimate concerns about their personal safety, if such information should fall into the wrong hands. Laws and rules of varying clarity, offering different degrees of protection, have been or are in the process of being enacted in the U. S., the European Union and Japan [2].
- **User Information Privacy:** Privacy of mobile users can be compromised by revealing the sensitive profile information of the mobile users to unintended users. The needs of mobile commerce applications go beyond tracking users' locations,

for example, they may additionally need to track user profiles and preferences in order to achieve mass personalization. This is because, to be effective, targeted advertising should not overwhelm the mobile consumers and must push information only to a certain segment of mobile consumers based on their preferences and profiles, and based on certain marketing criteria. Obviously, these consumers should be targeted only if they are in the location where the advertisement is applicable at the time of the offer. It is important to note here that user profile information may include both sensitive and non-sensitive attributes such as name, address, linguistic preference, age group, income level, marital status, education level, etc.

While mobile consumers like to benefit from personalization, they usually are not willing to share their sensitive profile information to all the merchants. To ensure the privacy of mobile users, it is important that the sensitive profile information is revealed to the respective merchants only on the need-to-know basis.

– **Security:** In addition to the privacy concerns mentioned above, there are a number of applications that call for securing resources based on the criteria of mobile objects. These include context (location)-sensitive access control, and ubiquitous computing environment, where access is permitted based on the location of the subjects/objects during a specific time.

In particular, the security policies provide controlled access to the mobile user profiles, to their current location and movement trajectories, to mobile resources, stationary resources based on the mobile user location. Therefore, an appropriate access control mechanism must be in place to enforce the authorization specifications reflecting the above security and privacy needs.

Access policies are specified as a set of authorizations, where each authorization states if a given subject possesses privileges to access an object. In the mobile environment, both subjects and objects can either be mobile or non-mobile. As a result either a subject or an object in an authorization specification can be a moving object. The access requests in such an environment can typically be on *past, present* and *future* status of the moving objects [26,13]. To effectively serve such access requests, one must efficiently organize the mobile objects as well as authorizations.

Although implementation of authorizations as access control list, capability list or access matrix is suitable for traditional data, it is not suitable to search authorizations in a mobile object environment, as they are based on spatial and temporal attributes of subjects and objects, rather than subject and object identifiers. Therefore, when a subject issues an access request, be it a past, future or current, the system must first retrieve the relevant object(s) from the moving object database, and then verify whether there exists an authorization that allows the subject to access these objects. Since both the moving objects and authorizations are spatiotemporal in nature, for efficient processing of access requests, it is essential that they both be organized using some index structures. As a result, processing an access request requires searching two indexes - one, the moving object index, and the other, the authorization index. To improve the response time of access requests, in this paper, we propose a unified index structure to index both moving objects and authorizations that govern access to them. Essentially, our index is created by carefully overlaying authorizations on top a moving object index, based on their spatiotemporal parameters.

Recently, a number of moving object index structures have been proposed. Unlike traditional spatiotemporal objects, moving objects are characterized by the moving spatial location that changes with time. In other words, a moving object can be specified as $\langle \bar{x}, \bar{v} \rangle$, where $\bar{x}$ represents its initial position vector and $\bar{v}$ its velocity vector. Current moving object index structures can be categorized primarily into three types: The first type stores the moving objects as transformed points in 2-dimensional dual $\langle \bar{x}, \bar{v} \rangle$ space [12,14], where dual transformation [15] is adopted. The second type stores them as lines in (d+1)-dimensional $\langle \bar{x}, t \rangle$ space (TB-tree and STR-tree [18,17]) by adding time t as an addition dimension. The third type stores them as points in native, d-dimensional $\langle \bar{x} \rangle$ space (TPR-tree [20]). Our proposed unified index structure, ^{S}TPR-tree, is constructed by carefully overlaying authorizations on top the TPR-tree, based on their spatiotemporal parameters. As a result of the unified index, access requests can be processed in one pass, thereby improving the response time. Note that current access control systems do not use any index for authorizations; our work is a step in this direction. We show how the ^{S}TPR-tree can be constructed and maintained, and provide algorithms to process access requests.

This paper is organized as follows. We first present the preliminaries of the TPR-tree in section 2. In section 3, we propose our moving object authorization model. In section 4, we present our proposed novel unified index structure, the ^{S}TPR-tree and illustrate our approach and strategy to overlay authorizations on top of the TPR-tree. In section 5, we describe how to process an access request that involves both searching for a moving object and evaluation of an authorization can be performed simultaneously. In section 6, we discuss the properties and limitations of our ^{S}TPR-tree. Related work is presented in section 7. We conclude the paper by providing some insight into our future research in this area in section 8.

2 Preliminaries of the TPR-Tree

In this section, we present the details of the TPR-tree[20] since our ^{S}TPR-tree is based on this. In particular, we present how a moving object is represented in the tree and how the tree is constructed to index these objects for efficient retrieval.

2.1 Representation of Moving Objects

Moving objects are data with attributes that change with time. Generally speaking, these objects may move in a d-dimensional embedding space. In this paper, for ease of visualization and explanation, we consider the space to be 2-dimensional. However, the formalism can be easily extended to higher dimensional spaces.

Let the set of moving objects be $MO = (mo_1, mo_2, \ldots mo_k)$. In the d-dimensional space, objects are specified as points which move with constant velocity $\bar{v} = \{v_1, v_2, \ldots v_d\}$ and initial location $\bar{x} = \{x_1, x_2, \ldots x_d\}$. The position $\bar{x}(t)$ of an object at future time $t (t \geq t_c)$ can be computed through the linear function of time, $\bar{x}(t) = \bar{x}(t_0) + \bar{v}(t - t_0)$ where t_0 is the initial time, t_c the current time and $\bar{x}(t_0)$ the initial position. Considering a two-dimensional space, a moving object mo_i moving in $\langle x, y \rangle$ space can be represented as follows: $mo_i = ((x_i, v_{i_x}), (y_i, v_{i_y}))$.

2.2 Time Parameterized Rectangle (*tpr*)

Given the trajectories of a set of moving objects MO in the time interval $[t_0, t_0 + \delta t]$ in $\langle x, y, t\rangle$ space, we define the tpr of MO as a 3-dimensional bounding trapezoid which bounds all the moving objects in MO during the entire time interval $[t_0, t_0 + \delta t]$. The tpr of MO can be defined as $(x^{\vdash}, x^{\dashv}, v_x^{\vdash}, v_x^{\dashv}, y^{\vdash}, y^{\dashv}, v_y^{\vdash}, v_y^{\dashv})$ and its projection on $\langle x, t\rangle$ space is a time-parameterized bounding interval $[x^{\vdash}(t), x^{\dashv}(t)] = [x^{\vdash}(t_0) + v_x^{\vdash}(t - t_0), x^{\dashv}(t_0) + v_x^{\dashv}(t - t_0)]$ and the projection on $\langle y, t\rangle$ space is another time-parameterized bounding interval $[y^{\vdash}(t), y^{\dashv}(t)] = [y^{\vdash}(t_0) + v_y^{\vdash}(t - t_0), y^{\dashv}(t_0) + v_y^{\dashv}(t - t_0)]$, where $\forall i \in \{1, 2, \ldots, k\}$

$$x^{\vdash} = x^{\vdash}(t_0) = min_i\{x_i(t_0)\} \qquad v_x^{\vdash} = min_i\{v_{i_x}\}$$
$$x^{\dashv} = x^{\dashv}(t_0) = max_i\{x_i(t_0)\} \qquad v_x^{\dashv} = max_i\{v_{i_x}\}$$
$$y^{\vdash} = y^{\vdash}(t_0) = min_i\{y_i(t_0)\} \qquad v_y^{\vdash} = min_i\{v_{i_y}\}$$
$$y^{\dashv} = y^{\dashv}(t_0) = max_i\{y_i(t_0)\} \qquad v_y^{\dashv} = max_i\{v_{i_y}\}$$

For example, figures 1 and 2 show a time parameterized rectangle and the trajectory of three moving objects a,b and c in $[t_0, t_0 + H]$ time interval, together with their projections on both $x-$ and $y-$ dimensions, respectively. We explain how this tpr can be constructed using figure 3, which depicts 3 moving objects, a,b and c in $\langle x, y, t\rangle$ space. Based on their respective velocities, the lines represented by a_x and c_x form the lower and upper bound in the x-dimension, and a_y and c_y in y-dimension, respectively. As a result, the tpr shown in the shaded region in figure 4 is formed. Note that the axes in figures 3 and 4 have been changed from those of figures 1 and 2 to improve readability. As shown in figure 1, at each time point, say t_0, t_i and $t_0 + H$ the the slice of the tpr is a rectangle with area A_0, A_i, and A_H, respectively.

The *tpr* hierarchy: Given a set of tprs, they can be organized in a hierarchical structure. As can be seen in figure 5, tpr C encloses tprs A and B. These three can be organized as a hierarchical structure with A and B being the children of C, as shown in figure 6. Essentially, at the bottom most level of the hierarchy, a set of moving objects could be grouped to form tprs. Each tpr of the next higher level is the bounding tpr of the set of tprs of all of its children. The root of the hierarchy is thus the bounding tpr covering all its lower level tprs in a recursive manner. Each tpr has exactly the same attributes as its children tprs except being larger in terms of its *spatial* magnitude while temporal span remains the same.

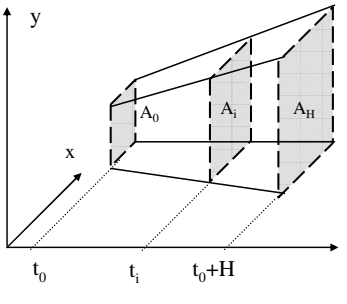

Fig. 1. TPR

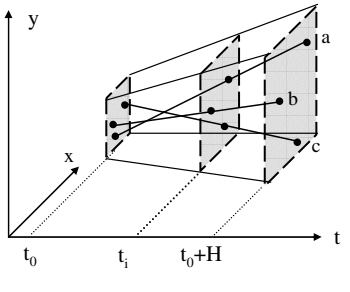

Fig. 2. The Trajectory of Moving Objects in TPR

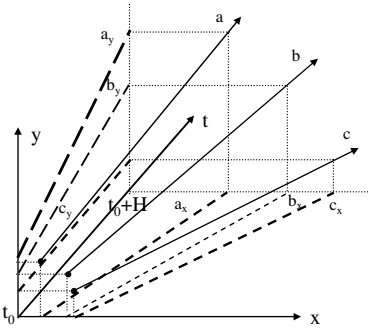

Fig. 3. The three moving objects a,b,c

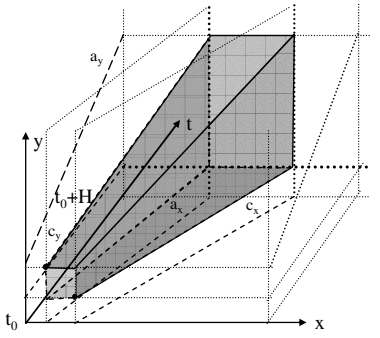

Fig. 4. The resultant tpr of figure 3

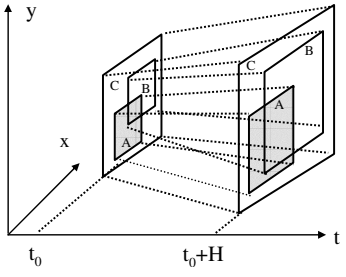

Fig. 5. The tpr Hierarchy

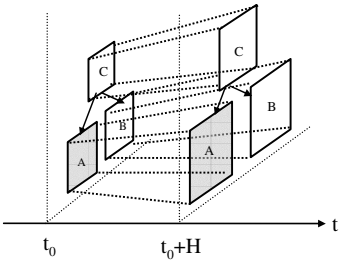

Fig. 6. The tpr hierarchy forming a tree

2.3 The TPR-Tree

In this section, we first introduce some essential parameters related to the TPR-tree.

Time Horizon (H): Recall that each moving object is represented by its initial position and velocity. However, given a moving object, it is unrealistic to assume that its velocity remains constant. Therefore, the predicted future location of a object specified as a linear function of time becomes less and less accurate as time elapses [20]. To address this issue, the TPR-tree defines a *time horizon*, H, representing the time interval during which the velocities of the moving objects hold good. It assumes that the tree is constructed by bulkloading the moving objects at some point in time (say t_0) and reconstructs the tree after H. In essence, the tree is good during $[t_0, t_0 + H]$ interval and all predictions made within this interval are acceptable in terms of the degree of accuracy. However, the tree deteriorates beyond its time horizon.

Construction of the TPR-Tree: TPR-tree is a time parameterized variant of R*-tree [10]. Each node in the TPR-tree represents a tpr of a set of moving objects. The way objects and in turn tprs are grouped and therefore be placed in a specific node is based on an *objective function* that can be specified using any of the following strategies, including the smallest sum of volumes of tprs, the smallest sum of the overlapping

regions among tprs, etc. One may adopt an appropriate strategy based on the application under consideration. So each node stores the bounding rectangle that grows with time to cover the enclosing moving objects. Thus, the bounding rectangle forms the 3-dimensional bounding trapezoid. While objects are stored only at the leaf nodes, the purpose of the upper level nodes is to direct the search via a specific path(s) to efficiently retrieve the objects that satisfy a user request.

The tree is constructed via the consecutive insertion operation into an initially empty tpr root. The candidate node to insert an object should either spatiotemporally enclose the new object being inserted, or needs the least volume enlargement to enclose this new object in time interval$[t_0, t_0 + H]$. The tpr of its ancestor node should be expanded to cover its children tprs, if necessary. The detailed steps for insertion are similar to that of R*-tree. We use $N^{\square}$ to denote the tpr of node N.

3 Moving Object Authorization Model

In this section, we propose an authorization model suitable for moving object data. An authorization, in general, is specified on the basis of three parameters, $\langle s, o, p\rangle$. This triple specifies that subject s is authorized to exercise privilege p on object o. Note that both subjects and objects can be a moving object. To avoid confusion, from now on, we denote the objects as *auth-objects* (stands for authorization objects) and *mov-objects* (stands for moving objects). Let $S = \{s_1, s_2 \ldots\}$ denote a set of subjects, and $O = \{o_1, o_2 \ldots\}$ a set of auth-objects, and MO a set of mov-objects. Note that in a moving object environment, authorization specifications should be capable of specifying access control policies based on spatiotemporal attributes of both subjects and auth-objects.

Example 1. Examples of policies include:

- **Policy 1:** *A mobile (phone/service) customer is willing to reveal his personal profile information to certain merchants only during the evening hours, and when he is close to the shopping mall. Note that, in this case, only the auth-object is a mov-object and this policy is based on object's spatiotemporal attributes.*
- **Policy 2:** *An employee may use print services only between "9am and 5pm" and while he is "in the office.' Note that in this case, while the subject is a mov-object, the auth-object is not. Also note that the policy is based on the subject's spatiotemporal attributes.*

In the following, we discuss the auth-objects, subjects, privilege modes, and define authorization specifications. We have defined the mov-objects in section 2.

Definition 1. [Authorization] A *authorization* α is a 4 tuple $\langle ce, ge, p, \tau\rangle$, where ce is a *credential expression* denoting a group of subjects, ge is a *object expression* denoting a set of auth-objects, p is a set of privilege modes, and τ is a temporal term. $\square$

In this paper, we assume the formalism developed in [6] to specify ce, ge and τ. Due to space limitations, we do not review the details. Essentially, ce can simply be a subject identifier(s) or an expression specified over the spatiotemporal attributes of subject

credentials. Similarly, ge can be an auth-object identifier or spatiotemporal attributes of the auth-object. τ can be a time point, a time interval or a set of time intervals.

As mentioned earlier, subjects can either be moving or static. For example, while the subject is static in policy 1, it is mobile in policy 2. In other words, the credential expression may involve attributes of the subject that are mobile in nature. for database access purpose. As an example, a police officer may exercise her privileges and issue a traffic violation ticket only if she is in her jurisdiction.

Auth-objects to be protected may include the traditional objects that are not mobile in nature, as well as attributes of the mov-objects. Again, some of the attributes of the moving objects are time dependent (rather mobile) such as the location or trajectory information of the mov-object, whereas certain other attributes are time-independent (e.g., the profile information of the person carrying a mov-object.)

The privilege modes include the traditional modes such as `read, write` and `update` as well as those specific to moving objects such as `locate` and `track`.

Example 2. Examples of policies include:

- α_1= $\langle$*merchant(i), {profile(i)* $\wedge$ *rectangle(j)=(50,60,10,10)* $\wedge$ *[5pm, 9pm]}, read* $\rangle$
- α_2= $\langle$*{Tom(i)* $\wedge$ *rectangle(j)=(50,60,10,10)* $\wedge$ *[9pm, 5pm]},{printer}, execute*$\rangle$

Given an authorization $\alpha = \langle ce, ge, p, \tau \rangle$, we use S^α to denote the set of subjects that satisfy ce, $x_b^\alpha, x_e^\alpha, y_b^\alpha$ and y_e^α to denote the spatial extent specified by ge represented by the lower and upper bounds in the x and y dimensions, respectively, and $[\tau_b^\alpha, \tau_e^\alpha]$ to denote the time interval during which α is valid. We use $\alpha^\square$ to denote the spatiotemporal region specified by the authorization α. Essentially, $\alpha^\square$ is nothing but the region specified by $x_b^\alpha, x_e^\alpha, y_b^\alpha, y_e^\alpha, \tau_b^\alpha$, and τ_e^α.

4 The ^{S}TPR-Tree

In this section, we present the ^{S}TPR-tree, which is a secure extension of the TPR-tree in which authorizations are carefully overlaid on the nodes of the TPR-tree based to allow efficient evaluation of access requests. Recall that the TPR-tree is valid only for a fixed duration of the time, H, since the location of the objects is constantly changing. In other words, the tree accurately represents the data values during H, but there are no guarantees on its accuracy that may deteriorate beyond H as the velocity of the mov-object recorded in the tree may no longer hold. While H is typically of a relatively of short duration, the time interval ($[\tau_b, \tau_e]$) associated with the authorizations that specifies the validity period of the authorizations, is much longer. Therefore, in order to overlay authorizations on the short-lived TPR-tree, we must slice each authorization based on H. Recall that α_τ can a timestamp or a single interval or multiple intervals. For the sake of simplicity, in the following, our discussion and formalism primarily focuses on a single time interval$[\tau_b, \tau_e]$. Our formalism can be easily extended to multiple intervals. Note that a timestamp can be represented as an interval $[\tau_b, \tau_b + 1]$.

Whenever $[\tau_b^\alpha, \tau_e^\alpha] > H$, we partition that α such that the time intervals of the partitions are mutually disjoint fragments whose interval does not exceed H.

Definition 2. [**Authorization Partition**] Given an authorization α and the time horizon H, the set of partitions of α, $\mathcal{P}^\alpha = \{\alpha_1, \alpha_2 \ldots \alpha_s\}$ such that $\forall \alpha_r \in \mathcal{P}^\alpha$, the following properties are satisfied:

1. $\tau^{\alpha_i} \cap \tau^{\alpha_j} = \emptyset, i \neq j$;
2. $[\tau_b^{\alpha_r}, \tau_e^{\alpha_r}] \leq H$;
3. $\cup_{r=1}^{s}[\tau_b^{\alpha_r}, \tau_e^{\alpha_r}] = H$; and
4. $\forall i, j \in \{1 \ldots s\}, i \neq j, x_b^{\alpha_i} = x_b^{\alpha_j}, x_e^{\alpha_i} = x_e^{\alpha_j}, y_b^{\alpha_i} = y_b^{\alpha_j}$ and $y_e^{\alpha_i} = y_e^{\alpha_j}$. □

Essentially, the above definition states that the time interval of each authorization is partitioned into intervals no greater than H and the union of the intervals of all partitions should be same as H. Moreover, authorization partitioning does not change the spatial extent of the authorization. From now on, we denote an authorization partition as an authorization since their structures are the same except for the difference in their time interval. Without loss of generality, from now on in this paper, we use authorizations to refer to their partitions since we are concerned about a single time horizon H at any given time.

4.1 Categorization of Authorizations

Given the spatiotemporal extent of the authorization ($\alpha^{\square}$) and the tpr of a node N in the tree ($N^{\square}$), we are interested in the following three possible cases: (i) the former fully contains the latter, (ii) the former overlaps with the latter and (iii) the former is disjoint with the latter. Based on these three cases, we determine whether to overlay an authorization on a specific node in the tree. Accordingly, we label the authorizations (or rather their partitions), as one of the four types: *enclosing authorization, overlapping authorization*, *disjoint authorization* and *pending authorization*.

Let the binary operators $\supset_{\{x,y,t\}}, \cap_{\{x,y,t\}}$ and $\otimes_{\{x,y,t\}}$ denote *enclose*, *overlap* and *disjoint*, respectively, in all x, y and t dimensions.

Definition 3. Given a tree T, node N in T, and an authorization α, we define,

- *Enclosing Authorizations*, $\alpha_E(N) = \{\alpha | \alpha^{\square} \supset_{\{x,y,t\}} N^{\square} = True\}$;
- *Overlapping Authorizations*, $\alpha_O(N) = \{\alpha | \alpha^{\square} \cap_{\{x,y,t\}} N^{\square} = True\}$;
- *Disjoint Authorizations*, $\alpha_P(N) = \{\alpha | \alpha^{\square} \otimes_{\{x,y,t\}} N^{\square} = True\}$; and
- *Pending Authorizations*, $\alpha_P(T) = \{\alpha | \forall N \in T\ (\alpha \in \alpha_P(N))\}$. □

For example, in figure 7, the TPR-tree consists of 3 nodes A, B and C with A and B being the children of C. Let there be 4 authorizations $\alpha_1, \alpha_2, \alpha_3$ and α_4. Each shaded area represents the intersecting region between an authorization and a tree node. As we can be seen, $\alpha_1 \in \alpha_O(C)$ and $\alpha_2 \in \alpha_O(B)$. Note that at the same time it could be possible that $\alpha_1 \in \alpha_P(A)$, if ${\alpha_1}^{\square}$ is disjoint with $A^{\square}$. Since α_4 is disjoint with $A^{\square}$, $B^{\square}$ and $C^{\square}$, $\alpha_4 \in \alpha_P(T)$.

4.2 Authorization Overlaying

In this section, we present our approach and strategy to overlay authorizations on top of the TPR-tree. The resultant tree is the ^{S}TPR-tree. Algorithm 1 presents the details of our spatiotemporal overlaying approach. An authorization is overlaid on a node if it is an enclosing authorization, that is, if it both spatially *and* temporally encloses the node's tpr. Essentially, it enforces the following two rules.

Authorization Overlaying Rules: Given a TPR-tree T, for *each* path from root node to a leaf node, it enforces the following two overlaying rules, Rule 1 and Rule 2. These rules are applied in a specific order, first Rule 1 and then Rule 2.

- **Rule 1:**
  - **a:** If the authorization $\alpha \in \alpha_E(N)$, that is, if its spatiotemporal extent completely encloses that of node N, overlay α on N and add it to the list of enclosed authorizations, $Auth_E(N)$. Halt the overlaying process of this authorization on this path.
  - **b:** If the authorization $\alpha \in \alpha_P(N)$, that is, if its spatiotemporal extent is disjoint with that of N, halt the overlaying process of this authorization on this path.
- **Rule 2:** If N is the leaf node, overlay α on N and add it to the list of enclosed authorizations, $Auth_O(N)$.

In the following, we provide the details of the various steps of our spatio-temporal overlaying algorithm. Essentially, algorithm 1 traverses the tree recursively in a top down manner starting from the root node. At each node N, $\alpha^{\square}$ is compared with $N^{\square}$. Several cases could occur based on the result of this comparison.

Case 1: If the authorization fully encloses the node, we will stop traversing the subtree of N and overlay α on that node, specifically in $Auth_E(N)$. This is because, it is reasonable to assume that, if a subject is allowed to access objects within a certain spatiotemporal region, it is always allowed to access objects in the *subregion* of that. Therefore, after overlaying an authorization on a node, there is no need to overlay the same authorization on any of its descendents. However, we still need to check if the authorization's spatiotemporal extent overlaps with that of any of its sibling nodes. Since the enclosing relation between α and N has no implication on the spatiotemporal relations between α and N's siblings, this step is essential.

Case 2: If the authorization's extent overlaps with that of the node, then we handle it differently depending on the level of the node in the tree.

- If the node is at the leaf level, then we overlay α and store it in $Auth_O(N)$ associated with that leaf node. This is because, when the authorization only overlaps with a leaf node, it means that it cannot fully enclose any node from root to that leaf in that path. However, to ensure that no relevant authorizations are discarded, this must be overlaid on the leaf node only.
- If the node is at the non-leaf level, then we traverse to the level below, and compare $\alpha^{\square}$ with *each* of the children nodes of N. This is because, the authorization may have different spatiotemporal relationships with each of its children nodes.

Case 3: If the spatiotemporal extent of the authorization is disjoint with the node, then we stop traversing the subtree and store α in $Auth_P(N)$ to be included in the pending authorizations of the tree. After each H interval, when a new TPR-tree is constructed based on the new position and velocity information of each moving object, the previously non-applicable authorizations may become relevant.
Based on this overlaying strategy, we can make the following observations:

- It is possible that an authorization may be overlaid on more than one node, resulting in a number of copies of the same authorization being stored in the tree. However, given a specific path from the root to a leaf, there may exist at most one copy of an authorization, regardless of its location or whether it is stored in $Auth_E(N)$ or $Auth_O(N)$.

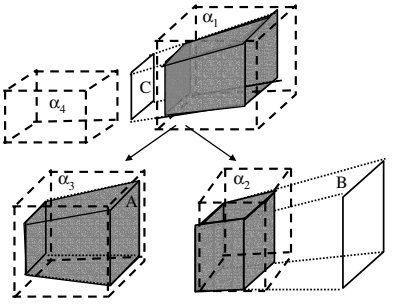

Fig. 7. Four categories of authorizations

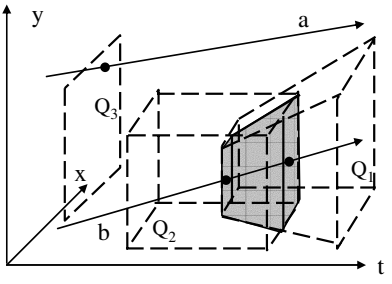

Fig. 8. The different query types

- Although the spatiotemporal extent of an authorization at node N may not completely enclose the spatiotemporal extents of the authorizations at the descendents of N, it may *cover* them. By cover we mean that, it is not necessary to further traverse the subtree if an authorization is found at that node. For example, if an authorization is overlaid on node C in the ^{S}TPR-tree in figure 7, it covers C's children nodes A and B for the same subject(s) in the overlaid authorization.
- The enclosing authorizations are overlaid on a node as high as possible in the tree. In other words, an authorization α will be overlaid on a node N if there exists no other node M at higher level than N in the tree such that $\alpha \in \alpha_E(M)$.
- Leaf nodes may be associated with two sets of authorizations, $Auth_E(N)$ and $Auth_O(N)$, whereas, non-leaf node have only one set of authorizations, $Auth_E(N)$, associated with them.

5 Access Request Evaluation

In this section, we present our approach to processing the access requests for different types of queries posed in the mobile object environment.

5.1 Query Types

The user request can fall into one of the following query types.

Definition 4. [**Generalized Query**] We define a generalized query, denoted as a quadruple $Q = \langle R_b, R_e, \tau_b, \tau_e \rangle$, which is a 3-dimensional trapezoid in the spatiotemporal space that has two parallel rectangular bases R_b and R_e projected to point τ_b and τ_e on the temporal dimension respectively.

There are two special cases of the generalized query, which are as follows:

- **Time Point Query:** is the generalized query such that $R_b = R_e = R, \tau_b = \tau_e = \tau$.
- **Time Interval Query:** is the generalized query such that $R_b = R_e = R$.

For example, figure 8 depicts the above three types of queries. Evidently, Q_1 is a generalized query, Q_2 is a time interval query and Q_3 is a time slice query. Note that, while the moving object a satisfies Q_3, the moving object b satisfies both Q_1 and Q_2.

5.2 User Request Evaluation

This section presents the user request evaluation, presented in algorithm 2, to retrieve the moving objects that satisfy the request. Formally, a user request is a tuple, $U = \langle s, Q \rangle$. We use $U^{\square}$ and s^U to denote the spatiotemporal extent and the subject of the access request U.

The spatiotemporal query evaluation presented in algorithm 2 is based on our spatiotemporal overlaying strategy. In the following, we explain the various steps of algorithm 2. It traverses the tree from the root down to each node on the leaf level. During this traversal, it compares the spatiotemporal extent of U with that of each node N.

1. If the spatiotemporal extent of user request either overlaps or fully contains that of the node, the authorizations in either $Auth_E(N)$ or $Auth_O(N)$ are evaluated. Essentially, it checks if there exists any relevant authorization for the subject issuing the access request. If the node is at the leaf level, $Auth_E(N)$ is first checked for relevant authorizations. If none is found, then $Auth_O(N)$ is checked. If the node is a non-leaf node, $Auth_E(N)$ is checked for relevant authorizations. If none is found, the traversal continues. If a relevant authorization is found, there is no need to check any authorization associated with N's descendents. Since we do not consider negative authorizations, the subject under consideration is allowed access to the spatiotemporal extent of the descendents of N.
2. If the spatiotemporal extent of user's access request is disjoint with that of the node N, we simply stop the search since we already know that no relevant authorizations can be found in N's subtree to satisfy the user request. However, the spatiotemporal extent of user's access request may still overlap or enclose with other nodes on the same level of the tree, thus we still need to check N's siblings.

Note that, in algorithm 2, we use HEIGHT$_{leaf}$ to denote the height of the tree, i.e. the length of the path from the root to a leaf. The operation $\cup_{\{x,y,t\}}$ is *not* a boolean operator; it produces the union of the spatiotemporal extents. Also, we assume the default value of the boolean variable *authorized* is FALSE, which means that by default subjects are *not* entitled access unless an explicitly specified relevant authorization is found. The operation EVALUATE( ) computes and returns the intersecting region among its parameters, it returns $\emptyset$ if the intersection among the parameters is empty EVALUATE() is a *overloading* function in that it can take different number of parameters. RETRIEVE( ) takes the spatiotemporal extents as the parameter and retrieves all moving objects falling into those spatiotemporal extents.

6 Discussion

In this section, we discuss the properties of our proposed ^{S}TPR-tree with respect to completeness and efficiency, as well as its limitations which are planned to be addressed as part of our future work.

6.1 Properties of ^{S}TPR-Tree

In this section, we demonstrate that our overlaying strategy and evaluation strategies are both *complete* and *efficient.*

Theorem 1. Given a set of authorizations A, Algorithm 1 overlays authorizations on the ^{S}TPR-tree in such a way that the evaluation of access request U by algorithm 2 is *complete*.

Proof sketch: By complete, we mean, given a user access request U by subject s, we evaluate every authorization relevant to s. Let this set of relevant authorizations be $\alpha(s^U)$. This can be proved in three steps. First, we can prove that our overlaying procedure (algorithm 1) overlays every authorization in $\alpha(s^U)$ on the *search path* of U. Second, we can prove that our evaluation procedure (algorithm 2) traverses every search path and therefore it evaluates all $\alpha(s^U)$. Third, we can prove that our halting strategies of traversal in both the algorithms do not violate the completeness principle. □

In addition to being complete our overlaying strategy is efficient in the sense it does not do any additional traversal than necessary. This is achieved as a result of the two of the following strategies we adopt during the overlay process: We overlay the authorizations on the first node encountered on the search path that totally encloses the region covered by the node. As a result, authorizations are overlaid as high up as possible in the tree. Since the search path is from root to the leaf, we will encounter the relevant authorizations as early as possible during the traversal. Our traversal essentially avoids evaluating (1) non-relevant authorizations as much as possible, and 2) relevant but redundant authorizations as much as possible.

6.2 Limitations of Our ^{S}TPR-Tree

Our proposed index structure is not capable of layering authorizations with privilege mode `track`. This is because, in order to entertain such privilege mode, it requires persistent storage of the moving object location information. As such it requires the index that maintains historical information (in addition to the current and future). We defer this to our future work.

Moreover, our index structure is capable of overlaying authorizations where either subjects or auth-objects are moving objects. However, it is not capable of overlaying authorizations where both subjects and auth-objects are moving objects. For example, it is not capable of supporting authorization policies similar to the one specified in the example below. This is because, subjects and auth-objects are represented by different nodes of the moving object index tree. As a result, supporting such authorization overlaying may require splitting the subject and auth-object components. We defer this to our future work.

– **Policy 3:** A manager can access the location of his employee (John) information between "9am and 5pm" and while the manager is "in the office." Note that both the subject (manager in this case) and auth-object (his employee) are mov-objects. This policy is based on the subject's as well as auth-object's spatiotemporal attributes.

7 Related Work

Recently, there has been a flurry of activity in developing indexing structures for moving object databases to effectively serve queries of *past, present* and *future* types [26,13]

on moving objects. Sistla et al. [23] propose a Moving Objects Spatio-temporal Model (MOST) to model object location as a linear function of time. A modification to R-trees to represent past trajectories of moving points as polylines has been proposed by Pfoser et al. [18]. Other work on indexing future trajectories include the PMR-Quadtrees [21,24] by Tayeb et al. that index the future linear trajectories of one-dimensional moving point objects as line segments. These methods do not seem to apply to more than one dimension. Kollios et al. [12] provide theoretical lower bounds for this indexing problem. Basch et al. [9] propose main-memory data structure (kinetic) for mobile objects, whose ideas are applied by Agarwal et al. [3] to external range trees [4]. The Time Parameterized R-trees (TPR-trees) [20], which is an extension of R*-trees, proposed by Saltenis et al., rather than attempt to continuously update moving object locations [26], represent objects as functions of time. Saltenis et al. claim that, similar to regular R-trees, the TPR-trees are capable of indexing points in any number of dimensions and are easily extended to accommodate objects that are not points. In [16], Palanis et al. extend the TPR-tree to accommodate past queries.

Atluri and Mazzoleni have proposed a unified index, called RMX-Quadtree [8], for geo-spatial data and authorizations that govern access to them. In [7], Atluri and Guo have proposed a unified index called STAR-Tree that can uniformly index both spatiotemporal objects and the authorizations. STAR-Tree relaxes several restrictive assumptions of RMX-Quadtree and therefore is more general. While the above two contributions are relevant but are limited to index data as well as authorizations specified on spatiotemporal data that is static in nature. The focus of this paper is to provide uniform indexing scheme for mobile data and the respective authorizations.

An index scheme for moving object data and user profiles has been proposed by Atluri et al. [5]. However, this does not consider authorizations. Beresford et al. [11,22] have proposed techniques that let users benefit from location-based applications, while preserving their location privacy. Mobile users, in general, do not permit the information shared among different location based services. Primarily, the approach relies on hiding the true identity of a customer from the applications receiving the user's location, by frequently changing pseudonyms so that users avoid being identified by the locations they visit. A system for delivering permission-based location-aware mobile advertisements to mobile phones using Bluetooth positioning and Wireless WAP Push has been developed [1]. An index structure has been proposed to index authorizations ensuring that the customer profile information be disclosed to the merchants based on the choice of the customers [27]. However, this provides separate index structures for data and authorizations, and therefore is not a unified index.

8 Conclusions and Future Work

Often, enforcing security incurs overhead, and as a result, may degrade the performance of a system. In this paper, we address this problem in the context of enforcing access control policies in a mobile data object environment. Although implementation of authorizations as access control list, capability list or access matrix is suitable for traditional data, it is not suitable to search authorizations as they are based on spatial and temporal attributes of subjects and objects, rather than subject and object identifiers.

Therefore, when a subject issues an access request, the system must first retrieve the relevant object(s) from the moving object database, and then verify whether there exists an authorization that allows the subject to access these objects. Since both the moving objects and authorizations are spatiotemporal in nature, for efficient processing of access requests, it is essential that they both be organized using some index structures. As a result, processing an access request requires searching two indexes - one, the moving object index, and the other, the authorization index. To improve the response time of access requests, in this paper, we have proposed a unified index structure, called ^{S}TPR-tree to index both moving objects and authorizations that govern access to them. Essentially, our index has been created by carefully overlaying authorizations on top a moving object index, specifically the TPR-tree. As a result of the unified index, access requests can be processed in one pass, thereby improving the response time. We have shown how the ^{S}TPR-tree can be constructed and maintained, and provided algorithms to process access requests.

Currently, we are conducting a performance evaluation to demonstrate that our uniform indexing scheme indeed has significant impact on the response time. Note that current access control systems do not use any index for authorizations. Our work is a step in this direction of improving the response time of access requests.

Our proposed unified indexing scheme is not capable of historical queries, which requires persistent storage of moving objects [16]. As a result, our ^{S}TPR-tree cannot overlay and evaluate authorizations with the privilege mode $track$. In addition, our overlaying strategy we have adopted in the ^{S}TPR-tree is not capable of overlaying authorizations in which both subject and authorization objects happen to be moving objects. We will enhance our ^{S}TPR-tree to address these issues. Support for negative authorizations require significant changes to the overlaying of authorizations as well as evaluating access requests. In this paper, we do not consider negative authorizations; we will extend our work to support negative authorizations.

References

1. L. Aalto, N. Gthlin, J. Korhonen, and T. Ojala. Bluetooth and wap push based location-aware mobile advertising system. In *Proceedings of the international conference on Mobile systems, applications, and services*, pages 49–58, 2004.
2. L. Ackerman, J. Kempf, and T. Miki. Wireless location privacy: A report on law and policy in the United States, the Europrean Union, and Japan. *DoCoMo USA Labs Technical Report DCL-TR2003-001*, 2003.
3. P. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 175–186, 2000.
4. L. Arge, V. Samoladas, and J. Vitter. On two-dimensional indexability and optimal range search indexing. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 346–357,1999.
5. V. Atluri, N. Adam, and M. Youssef. Towards a unified index scheme for mobile data and customer profiles in a location-based service environment. In *Workshop on Next Generation Geospatial Information (NG2I'03)*, 2003.
6. V. Atluri and S. Chun. An authorization model for geospatial data. *IEEE Trans. Dependable Sec. Comput.*, 1(4):238–254, 2004.

7. V. Atluri and Q. Guo. STAR-Tree: An index structure for efficient evaluation of spatiotemporal authorizations. In *IFIP TC11/WG 11.3 Eighteenth Annual Conference on Data and Applications Security*, pages 31–47, 2004.
8. V. Atluri and P. Mazzoleni. Uniform indexing for geospatial data and authorizations. In *DBSec*, pages 207–218, 2002.
9. J. Basch, L. Guibas, C. Silverstein, and L. Zhang. A practical evaluation of kinetic data structures. In *Proceedings of the annual symposium on Computational geometry*, pages 388–390, 1997.
10. N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: an efficient and robust access method for points and rectangles. In *SIGMOD Conference*, pages 322–331, 1990.
11. A. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *PerCom Workshops*, pages 127–131, 2004.
12. G. Kollios, D. Gunopulos, and V. Tsotras. On indexing mobile objects. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 261–272,1999.
13. J. Moreira, C. Ribeiro, and T. Abdessalem. Query operations for moving objects database systems. In *Proceedings of the eighth ACM international symposium on Advances in geographic information systems*, pages 108–114. ACM Press, 2000.
14. D. Papadopoulos, G. Kollios, D. Gunopulos, and V. Tsotras. Indexing mobile objects on the plane. In *DEXA Workshops*, pages 693–697, 2002.
15. D. Papadopoulos, G. Kollios, D. Gunopulos, and V. Tsotras. Indexing mobile objects using duality transforms. *IEEE Data Eng. Bull.*, 25(2):18–24, 2002.
16. M. Pelanis, S. Saltenis, and C. Jensen. Indexing the past, present and anticipated future positions of moving objects. *A TIMECENTER Technical Report TR-78*, 2004.
17. D. Pfoser. Indexing the trajectories of moving objects. *IEEE Data Eng. Bull.*, 25(2):3–9, 2002.
18. D. Pfoser, C. Jensen, and Y. Theodoridis. Novel approaches in query processing for moving object trajectories. In *Proceedings of 26th International Conference on Very Large Data Bases*, pages 395–406,2000.
19. A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *MOBICOM*, pages 96–108, 2003.
20. S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, volume 29, pages 331–342, 2000.
21. H. Samet and M. Tamminen. Approximating csg trees of moving objects. *The Visual Computer*, 6(4):182–209, 1990.
22. D. Scott, A. Beresford, and A. Mycroft. Spatial security policies for mobile agents in a sentient computing environment. In *FASE*, pages 102–117, 2003.
23. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K*, pages 422–432, 1997.
24. J. Tayeb, O. Ulusoy, and O. Wolfson. A Quadtree-Based Dynamic Attribute Indexing Method. *The Computer Journal*, 41(3):185–200, 1988.
25. V. Venkatesh, V. Ramesh, and A. Massey. Understanding usability in mobile commerce. *Commun. ACM*, 46(12):53–56, 2003.
26. O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *10th International Conference on Scientific and Statistical Database Management, Proceedings, Capri, Italy, July 1-3, 1998*, pages 111–122, 1998.
27. M. Youssef, N. Adam, and V. Atluri, Preserving Mobile Customer Privacy: An Access Control System for Moving Objects and Customer Information, *International Conference on Mobile Data Management*, Lecture Notes in Computer Science, 2005.

A Algorithms

Algorithm 1. Spatiotemporal Overlay

Input: tree root N, authorization to be overlaid α, tree height h
Output: none
Assumption: the height of the root is 0
Constant Variable: HEIGHT$_{leaf}$
Initialization:$h = 0$
Procedure ST-Overlay (N, α, h)
if $(h =$ HEIGHT$_{leaf})$ **then**
  if $(\alpha^{\square} \supset_{\{x,y,t\}} N^{\square}) =$TRUE **then**
    $Auth_E(N) \leftarrow \alpha$
  else if $(\alpha^{\square} \cap_{\{x,y,t\}} N^{\square}) =$TRUE **then**
    $Auth_O(N) \leftarrow \alpha$
  else
    $Auth_P \leftarrow \alpha$
  end if
  return
end if
if $(\alpha^{\square} \supset_{\{x,y,t\}} N^{\square}) =$TRUE **then**
  $Auth_E(N) \leftarrow \alpha$
  return
end if
for each child C_i of N **do**
  ST-Overlay$(C_i, \alpha, h + 1)$
end for

Algorithm 2. Spatiotemporal Query Evaluation

Input: tree root N,user query U, tree height h
Output: the identifiers of targeted objects
Assumption: the height of the root is 0
Constant Variable: HEIGHT$_{leaf}$
Initialization:$h = 0, authorized =$ FALSE
Procedure ST-QueryEval$(N, U, h, authorized)$
if $(U^{\square} \otimes_{\{x,y,t\}} N^{\square}) =$ TRUE **then**
  return $\emptyset$
end if
$overlap \leftarrow \emptyset$
if $authorized =$ FALSE **then**
  if $\exists \alpha \in Auth_E(N)((s^U \in S^{\alpha}) \wedge ((U^{\square} \cap_{\{x,y,t\}} \alpha^{\square}) =$TRUE $))$ **then**
    $authorized \leftarrow$ TRUE
  end if
  if $(authorized =$ FALSE $) \wedge (h =$HEIGHT$_{leaf})$ **then**
    for each $\alpha \in Auth_O(N)$ **do**
      if $s^U \in S^{\alpha}$ **then**
        $overlap \leftarrow overlap \cup_{\{x,y,t\}}$ EVALUATE $(U^{\square}, N^{\square}, \alpha^{\square})$
      end if
    end for
    return RETRIEVE $(overlap)$
  else if $h =$HEIGHT$_{leaf}$ **then**
    $overlap \leftarrow$ EVALUATE$(U^{\square}, N^{\square})$
    return RETRIEVE$(overlap)$
  end if
else if $h =$HEIGHT$_{leaf}$ **then**
  $overlap \leftarrow$ EVALUATE$(U^{\square}, N^{\square})$
  return RETRIEVE$(overlap)$
end if
for each child C_i of N **do**
  ST-QueryEval$(C_i, U, h + 1, authorized)$
end for