Efficient Security Policy Enforcement in a Location Based Service Environment*

Vijayalakshmi Atluri and Heechang Shin

MSIS Department and CIMIC, Rutgers University, USA
{atluri,hshin}@cimic.rutgers.edu

Abstract. Location based services, one of the promising markets of mobile commerce, aims at delivering point of need personalized information. Often, these services to be delivered are based on the prior knowledge of the profiles of mobile customers and security and privacy policies dictated by them. These policies may specify revealing the sensitive information of mobile customers (e.g., age, salary) selectively to specific merchants in return of receiving certain benefits (e.g., coupons, special discounts, etc.). As a result, the security policies in such an environment are characterized by spatial and temporal attributes of the mobile customers (location and time), as well as their profile attributes. The focus of this paper is to efficiently enforce such policies. In this regard, we propose a unified structure that is capable of indexing mobile customer (mobile object) locations and their profiles, and the authorizations stating their security and privacy policies.

1 Introduction

In recent years, mobile phones and wireless PDAs have evolved into wireless terminals that are Global Positioning System (GPS) enabled. With the expected revenues of mobile commerce to exceed $88 billion by 2009 [12], mobile commerce will soon become a gigantic market opportunity. The market for location-aware mobile applications, often known as *location-based services* (LBS), is very promising. LBS is to request usable, personalized information delivered at the point of need, which includes information about new or interesting products and services, promotions, and targeting of customers based on more advanced knowledge of customer profiles and preferences, automatic updates of travel reservations, etc. For example, a LBS provider can be designed to present users with targeted content such as clothing items on sale, based on prior knowledge of their profile, preferences and/or knowledge of their current location, such as proximity to a shopping mall [13]. Additionally, LBS can provide nearby points of interests based on the real-time location of the mobile customer, advising of current conditions such as traffic and weather, deliver personalized, location-aware, and context-sensitive advertising, again based on the mobile customer profiles and preferences. Whether such LBS is delivered in a "push" or "pull" fashion, service providers require access to customers' preference profiles either through a proprietary database or use an arrangement with an LBS provider, who matches customer profiles to vendor offerings [11].

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

S. Barker, G.-J. Ahn (Eds.): Data and Applications Security 2007, LNCS 4602, pp. 61–76, 2007.

In order to implement such services, customization and personalization based on the location information, customer profiles and preferences, and vendor offerings are required. 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. Certain segment of mobile consumers are willing to trade-off privacy by sharing such sensitive data with selective merchants, either to benefit from personalization or to receive incentives offered by the merchants. Therefore, it is important that the sensitive profile information is revealed to the respective merchants only on the need-to-know basis. For example, a security policy may specify that a customer is willing to reveal his age in order to enjoy a 20% discount coupon offered on sports clothing. But he is willing to do this only during the evening hours and while close to the store. As a result, the security policies in such an environment are characterized by spatial and temporal attributes of the mobile customers (location and time), as well as their profile attributes.

A trusted LBS service provider can ensure that the customer's security policy is enforced without revealing the real identity of the customer to the merchant. Thus, an appropriate access control mechanism must be in place to enforce the authorization specifications reflecting the above security and privacy needs.

Traditionally, access policies are specified as a set of authorizations, where each authorization states if a given subject possesses privileges to access an object. Considering the basic authorization specification $\langle subject, object, privilege \rangle$, in a mobile environment, a moving object can be a subject, an object, or both. Access requests in such an environment can typically be on *past, present* and *future* status of the moving objects [2,3]. Serving an access request requires to search for the desired moving objects that satisfy the query, as well as enforce the security policies. The focus of this paper is to address the problem of efficiently enforcing such security policies.

Enforcing security policies often degrades the performance of a system. One way to alleviate the problem is to efficiently organize the mobile objects as well as authorizations. An index scheme for moving object data and user profiles has been proposed by Atluri et al. [7], but this does not consider authorizations. Recently, Atluri and Guo [4] have proposed a unified index structure called ^{S}TPR-tree in which authorizations are carefully overlaid on a moving object index structure (TPR-tree) [6], based on their spatiotemporal parameters. One main limitation of the ^{S}TPR-tree is that it is not capable of maintaining past information. As a result, it cannot support queries based on past location and security policies based on $tracking$ of mobile users. More recently, Atluri and Shin [5], present an index structure, called S^{PPF}-tree, which maintains past, present and future positions of the moving objects along with authorizations by employing the *partial persistent storage*. 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 [1]. However, this provides separate index structures for data and authorizations, and therefore is not a unified index. In essence,

none of the previously proposed unified indexing schemes support security policy enforcement based on the profiles of the mobiles users.

In this paper, we present an index structure, called S^{STP}-tree, which maintains authorizations along with present and future locations as well as profiles of moving objects. In particular, we build on the concepts of the TPR-tree [6] and modify the node structure of the tree to hold profile information. Then, authorizations are overlaid suitably on the nodes of the tree. In order to support the profile information, we propose a Profile Bounding Vector (PV^B) which works similar to Minimum Bounding Rectangle (MBR) in the R-tree family. We demonstrate how the S^{STP}-tree can be constructed and maintained, and provide algorithms to process access requests. Our analysis shows that under normal circumstances, S^{STP}-tree performs better than utilizing two separate indexes (one for moving objects and another for profile). More specifically, if the data size of a PV^B is small enough so that the minimum number of the data (or children) that a leaf node (or non-leaf node) is the same between the S^{STP}-tree and the separate index structures, the analysis shows that our tree performs better.

This paper is organized as follows. Section 2 introduces preliminaries such as user profiles and the TPR-tree. In section 3, moving object authorization model is presented, and in section 4, we present our approach, called S^{STP}-tree, and section 5 illustrates our approach and strategy to evaluate user requests. Also, theoretical analysis is presented. In section 6, we conclude the paper by providing some insights into our future research in this area.

2 Preliminaries

In this section, we present the preliminary concepts on building an index structure for moving objects and profiles.

2.1 Moving Objects

Representation of Moving Objects: Let the set of moving objects be O=$\{o_1,o_2,\ldots,o_k\}$. 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 o_i moving in $\langle x, y\rangle$ space can be represented as follows: $o_i = ((x_i, v_{i_x}), (y_i, v_{i_y}))$.

Time Parameterized Bounding Rectangle ($tpbr$): Given a set of moving objects O = $\{o_1, \ldots, o_n\}$ in the time interval $[t_0, t_0 + \delta t]$ in $\langle x, y, t\rangle$ space, the $tpbr$ of O is a 3-dimensional bounding trapezoid which bounds all the moving objects in O during the entire time interval $[t_0, t_0 + \delta t]$ in the following way:

$$tpbr(O) = \{(x^{\vdash}, x^{\dashv}, y^{\vdash}, y^{\dashv}), (v_x^{\vdash}, v_x^{\dashv}, v_y^{\vdash}, v_y^{\dashv})\}\ where\ \forall\, i \in \{1, 2, \ldots, n\}$$

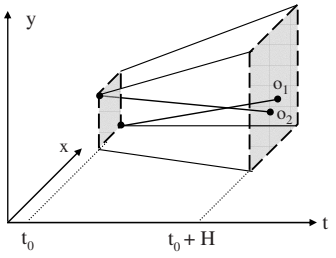

Fig. 1. The Time Parameterized Bounding Rectangle ($tpbr$)

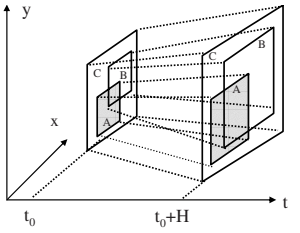

Fig. 2. The $tpbr$ Hierarchy

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

Then, we can compute the bounding rectangles that $tpbr$ covers with respect to time. The bounding rectangle's x-axis interval and y-axis interval at time t are defined as $[x^{\vdash}(t), x^{\dashv}(t)] = [x^{\vdash}(t_0) + v_x^{\vdash}(t - t_0), x^{\dashv}(t_0) + v_x^{\dashv}(t - t_0)]$ and $[y^{\vdash}(t), y^{\dashv}(t)] = [y^{\vdash}(t_0) + v_y^{\vdash}(t - t_0), y^{\dashv}(t_0) + v_y^{\dashv}(t - t_0)]$ respectively.

Time Horizon (H): 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 [6]. To address this issue, a *time horizon* H is defined, which represents the time interval during which the velocities of the moving objects assumed to be the same. Figure 1 shows how $tpbr$ bounds the trajectory of two moving objects o_1 and o_2 in $[t_0, t_0 + H]$.

The Tree Structure: Given a set of $tpbr$s, they can be organized in a hierarchical structure. In figure 2, $tpbr$ C encloses $tpbr$s A and B. These three can be organized as a hierarchical structure with A and B being the children of C, Essentially, at the bottommost level of the hierarchy, a set of moving objects could be grouped to form $tpbr$s. Each $tpbr$ of the next higher level is the bounding $tpbr$ of the set of $tpbr$s of all of its children. The root of the hierarchy is thus the bounding $tpbr$ covering all its lower level $tpbr$s in a recursive manner.

2.2 User Profiles

We assume user profile as a set of attributes associated with a mobile customer that characterizes the user. These attributes may include (1) demographic information (e.g. country, race, age, gender, etc.), (2) contact information (e.g., name, address, zip code,

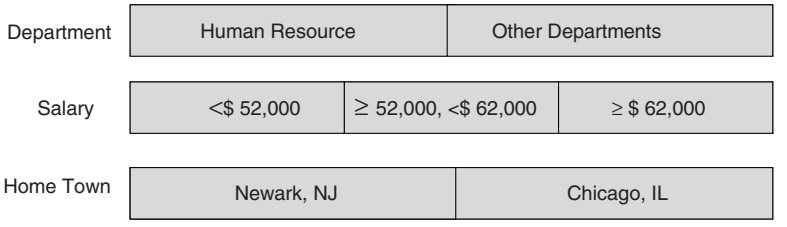

Fig. 3. Profile Attribute Discretization

telephone number, e-mail, etc.), (3) personal preferences (e.g., hobbies, favorite activities, favorite magazines, etc.), and (4) behavioral profile (e.g., level of activity, type of activity, etc.)[1]

Let the set of profile attributes under consideration be $P = \{p_1, p_2, \ldots, p_n\}$. We assume the profile of each user u is $\{p_1 : val_1, p_2 : val_2, \ldots, p_n : val_n\}$, where val_i is the corresponding value of p_i for that user. Since all attributes are not applicable to all users, some of these attributes may be empty for certain users.

Representation of User Profile: Given a profile attribute p_i, we first discretize it if necessary. We simply use as many bits as the number of all the possible discrete values for p_i. If the attribute is numerical data type, we partition the continuous data space into disjoint, mutually-exclusive intervals, as shown for attribute age in figure 3. The details of discretization method can be found in [14].

Definition 1 (Profile Vector). *Given a profile of user* $u = \{p_1 : val_1, p_2 : val_2, \ldots, p_n : val_n\}$*, we define a profile vector of u, denoted as* pv_u *as follows:* $pv_u = \{v_1, v_2, \ldots v_n\}$*, where each* v_i *is a sequence of binary digits such that the number of digits is equal to the number of discrete values of* p_i*, and the digits is 1 if* val_j *satisfies the corresponding discrete value, and 0 otherwise.*

Table 1. User Profile Information

Name	*Department*	*Salary*	*Home Town*	*Profile Vector*
Doe	Human Resource	$63,000	Newark, NJ	$\langle 10, 001, 10\rangle$
James	Other Departments	$45,000	Chicago, IL	$\langle 01, 100, 01\rangle$
Robert	Human Resource	$53,000	Chicago, IL	$\langle 01, 010, 01\rangle$

Figure 3 presents how each profile attribute can be represented. For example, because all the possible values of a profile attribute, $Department$ are two ("Human Resource" and

[1] The behavioral profile is created by observing activities and habits of a user continuously. For example, Sony TiVo box records frequently-watched television shows and generates a behavioral profile based on the past patterns. In order to do so, information such as what kind of activity has been done by a user at what intensity needs to be captured. In case of TiVo, type of activity can be 'watching drama' and level of activity can be '2 hours.'

"Other Departments"), we use two bits to represent $Department$: "Human Resource" is represented with '10' and "Other Departments" is represented with '01.' Also, if $Salary <$ \$52,000, we represent it with '100', and '001' if $Salary \geq$ \$62,000. Table 1 shows the examples of user profile vectors. For example, profile representation of the user, Doe, is $\langle 10, 001, 10\rangle$ because his department 'Human Resource', is represented as '10', salary, \$63,000, as 001, and home town, 'Newark, NJ' as '10'.

3 Moving Object Authorization Model

In this section, we review the authorization model presented in [4] that is capable of specifying access control policies based on the spatial and temporal attributes, and suitably extend it to specify policies based on the profile information of mobile users.

Definition 2 (Authorization). *An* authorization α *is a 4 tuple* $\langle se, ge, m, \tau\rangle$*, where* se *is a* subject expression *denoting a set of subjects,* ge *is a* object expression *denoting a set of objects,* m *is a set of privilege modes, and* τ *is a temporal term.* □

The formalism to specify se[2], ge and τ has been developed in [8]. Due to space limitations, we do not review these details in this paper. Subject expression se can be used to specify a set of subjects such that they are associated with (i) a set of spatiotemporal and/or other traditional credential attributes and/or profile attributes, (ii) a set of subject identifiers, or (iii) a combination of both. In the same way, object expression ge can be used to specify a set of objects such that they are associated with (i) a set of spatiotemporal and/or other types of attributes, (ii) a set of object identifiers, or (iii) a combination of both.

Note that the set of subjects and objects denoted by se and ge can be moving objects. Because both a subject and an object can be a moving object, to avoid confusion, from now on, we denote the objects specified in the authorization as *auth-objects* (stands for authorization objects). In a LBS environment, generally subjects are moving objects, and location-aware information (such as near-by restaurants, route management, and so on) is provided to subjects based on their location. A policy may state that a (moving) subject may access an auth-object such as files, printers based on the subject's spatiotemporal and profile attributes. Additionally, in a security policy, the (stationary) subject is allowed to access the profile information associated with a (moving) object.

The privilege mode m[3] supports not only read, write, and execute privileges for traditional auth-objects but also viewing, locating, tracking, send_sms_message for moving objects. A temporal term τ can be a time point, a time interval or a set of time intervals.

For a given authorization $\alpha = \langle se, ge, m, \tau\rangle$, we denote subjects expressed by se as $\alpha.se$, objects expressed by ge as $\alpha.ge$, privileges as $\alpha.m$, and temporal term $\alpha.\tau$, respectively.

There are two types of data associated with a moving object: spatiotemporal data (such as location) and non-spatiotemporal data (such as the profile information). Profile

[2] In [8], ce was used instead of se.

[3] Typically, the set of privileges M forms a partial order, where the ordering relationship can be represented with $\prec_m$.

information denotes a set of attributes that describes a moving object. For example, a person with handheld tracking devices can be described by her name, age, salary, occupation, and so on. Therefore, the security policies must be able to specify under what conditions, a mobile user wants to reveal her sensitive information. As such, given an authorization α, a subject expression se and an object expression ge, it is possible that the attributes involved in both se and ge could either be spatiotemporal in nature or not. Therefore, we extract the spatiotemporal and non-spatiotemporal part of the expression in α and denote them as $\alpha^{\square}$ and $\alpha^{\rightarrow}$, respectively. Note that, not all profile attributes of a mobile customer are relevant. We consider only those attributes that are involved in the entire set of policies to be included in the profile vector.

If se includes the specification of spatiotemporal region, it means that the moving subjects specified in se must be within the region to gain access to the objects specified in ge. Similarly, only the mobile objects in se that are within the region are allowed to be accessed by the subjects specified in se, if ge includes the specification of spatiotemporal region. Thus, an authorization embeds a spatiotemporal region for specifying authorizing conditions for se or ge.

In the following, we present some examples of security policies.

- Policy 1: In order to get a personalized promotion deal, a mobile customer is willing to reveal her age and salary information to a merchant, provided she is within 10 miles from the shopping mall during evening hours. This policy can be expressed as follows.
  α_1 = $\langle$merchant(i), {customer(j) $\wedge$ location(j)=circle((50,60),10) $\wedge$ [5pm, 9pm] $\wedge$ age(j) $\wedge$ salary(j) }, sms_message $\rangle$
  Here, only the attributes involved in ge are spatiotemporal in nature, but not those involved in se. In the above, $\alpha_1^{\square}$ is represented by a circle centered at (50,60) with radius 10 miles and t-axis interval = [5pm, 9pm]. Also, $\alpha_1^{\rightarrow} = \langle **, ***, ** \rangle$ because policy 1 does not evaluate the profiles of mobile users.
- Policy 2: When a mobile user enters the Newark Liberty International Airport, taxi companies are allowed to access her current location information if she is within the airport during office hours.
  α_2 = $\langle$taxi_company(i), {customer(j) $\wedge$ location(j)=rectangle(10, 70, 2, 2) $\wedge$ [9am, 5pm] }, locate$\rangle$[4]
  Here, only the attributes involved in ge are spatiotemporal in nature. $\alpha_2^{\square}$ is represented by a three dimensional rectangle with x-axis interval = [10, 12], y-axis interval = [70, 72], and t-axis interval = [9am, 5pm]. $\alpha_2^{\rightarrow} = \langle **, * * *, 01 \rangle$ because α_2 evaluates only the profile attribute 'Home Town'.
- Policy 3: Any employee can send a print job if she is currently located at the office during the office hours.
  α_3 = $\langle$emp(i) $\wedge$ rectangle(i)=(3,5,1,2) $\wedge$ [9am, 5pm], printer(j), write $\rangle$
  Here, only the attributes involved in se are spatiotemporal in nature, but not those involved in ge. $\alpha_3^{\square}$ is represented by a three dimensional rectangle with x-axis

[4] Services such as the *gazetteer service* [9] that convert canonical geographic area names into geo-coordinates are available today. For example, a place name such as "Newark Airport, NJ" can be converted into the coordinates as shown above.

interval = [3, 4], y-axis interval = [5, 7], and t-axis interval = [9am, 5pm]. Also, $\alpha_3^{\rightarrow} = \langle **, ***, ** \rangle$ because policy 3 does not evaluate the profiles of mobile users.

– Policy 4: A human resource employee is allowed to access performance records of employees only during office hours and while he is physically in his office.
α_4= $\langle$ emp(i) $\wedge$ department(i) = 'human_resource' $\wedge$ rectangle(i)=(3,5,1,2) $\wedge$ [9am, 5pm] $\}$, $\{$performance_record(j)$\}$, read $\rangle$
Only the attributes involved in se are spatiotemporal in nature, but not those involved in ge. $\alpha_4^{\square}$ is represented by a three dimensional rectangle with x-axis interval = [3, 4], y-axis interval = [5, 7], and t-axis interval = [9am, 5pm]. Also, $\alpha_4^{\rightarrow} = \langle 10, ***, ** \rangle$ because α_4 evaluates only the profile attribute 'Department'.

One main characteristic of authorization in the mobile environment is that the corresponding subject and object for se and ge are dynamically defined based on their location. For example, in a given authorization α, if $\alpha.ge$ specifies a region of Newark, NJ, $\alpha.se$ is authorized to access all the moving objects that lie within Newark, NJ to perform the privilege $\alpha.m$ on them. Thus, as time elapses, the list of authorized objects are being changed depending their movement inwards or outwards to the authorized area.

4 Unified Index Scheme for Moving Objects

In this section, we introduce our novel unified index structure, called the S^{STP}-tree that supports efficient enforcement of security policies based on present user locations as well as profiles. S^{STP}-tree is a balanced tree. Each node in the S^{STP}-tree comprises of the spatiotemporal attributes as well as a profile bounding vector, denoted as PV^B (explained later), in order to support the profile conditions. PV^B works similar to MBR (Minimum Bounding Rectangle) in R-tree. MBR in R-tree works as a coarse spatial filter that is used as a pre-filter to perform a more computationally expensive overlapping polygon checking [10]. Similarly, the role of profile bounding vector is to filter out profile conditions that do not satisfy the designated profile query conditions.

Profile Bounding Vector: In the following, we define a bounding vector that covers a set of profile vectors belonging to a set of users.

Definition 3 (Profile Bounding Vector). *Given a set of profile vectors* $PV = \{pv_1, pv_2, \ldots pv_n\}$, *such that each* $pv_i = \{v_{i1}, v_{i2}, \ldots v_{im}\}$. *A profile bounding vector of* PV, *denoted as* $PV^B = \{\{v_{11} \vee v_{21} \ldots v_{n1}\}, \{v_{12} \vee v_{22} \ldots v_{n2}\}, \ldots \{v_{1m} \vee v_{2m} \ldots v_{nm}\}\}$.

Let us consider once again the example in section 3 to explain the concept of PV^B. The set of profile attributes is department, salary, and home town. Consider the three profile vectors, $pv_{\text{Doe}} = \langle 10, 001, 10 \rangle$, $pv_{\text{James}} = \langle 01, 100, 01 \rangle$, and $pv_{\text{Robert}} = \langle 01, 010, 01 \rangle$. Then, PV^B of two users, Doe and James is $\langle 11, 101, 11 \rangle$, and PV^B of all three users is $\langle 11, 111, 11 \rangle$.

Given a set of PV^Bs, hierarchical structure can be formed. Suppose we have three PV^Bs.

$$PV_1^B = \langle 11, 011, 10 \rangle$$
$$PV_2^B = \langle 10, 010, 10 \rangle$$
$$PV_3^B = \langle 01, 001, 10 \rangle$$

These PV^Bs can be organized in a hierarchical structure with PV_2^B and PV_3^B as the children of PV_1^B. Each PV^B bounds PV^Bs of all of its children. Therefore, the root of the hierarchy covers the set of PV^Bs of all of its descendants.

Construction of S^{STP}-Tree: S^{STP}-tree is constructed similar to that of the tree structure described in section 2.1, but PV^B is updated accordingly during the insertion of new objects. As discussed in section 2, each moving object is represented with its spatiotemporal and profile attributes. Thus, each node in the S^{STP}-Tree includes a $tpbr$ and a PV^B for specifying the spatiotemporal and profile conditions, respectively. When a new moving object is being inserted into S^{STP}-tree, the first operation is to find a target leaf node that enlarges the $tpbr$ of the node smallest among all the leaf nodes. After inserting the object into the target leaf node, we update $tpbr$ and PV^B of the target leaf node if necessary. If $tpbr$ or PV^B of the parent node does not enclose all of its children as a result of inserting a new object into the target leaf node, we update them accordingly in the parent node. The same operation is applied to its parent node until the root node is reached recursively.

Relationships Between Authorization and Node: Given an authorization α and a node N, we are interested in different cases of spatiotemporal and PV^B relationships between α and N.

- *Spatiotemporal Relationship*
  - $\alpha^{\square} \supset_{st} N^{\square}$*:* spatiotemporal extent of α encloses that of N.
  - $\alpha^{\square} \cap_{st} N^{\square}$*:* spatiotemporal extent of α overlaps with that of N.
  - $\alpha^{\square} \otimes_{st} N^{\square}$*:* spatiotemporal extent of α is disjoint with that of N.
- *Profile Bounding Vector Relationship*
  - $\alpha^{\rightarrow} \supset_p N^{\rightarrow}$*:* $\alpha^{\rightarrow}$ encloses $N^{\rightarrow}$ if for each non-zero profile attribute vector[5] of $\alpha^{\rightarrow}$ and $N^{\rightarrow}$, bitwise 'OR' operation of $\alpha^{\rightarrow}$ and $N^{\rightarrow}$ results in $\alpha^{\rightarrow}$.
  - $\alpha^{\rightarrow} \cap_p N^{\rightarrow}$*:* $\alpha^{\rightarrow}$ overlaps with $N^{\rightarrow}$ if for each non-zero profile attribute of $\alpha^{\rightarrow}$ and $N^{\rightarrow}$, their bitwise 'AND' operation results in a non-zero profile attribute vector.
  - $\alpha^{\rightarrow} \otimes_p N^{\rightarrow}$*:* $\alpha^{\rightarrow}$ is disjoint with $N^{\rightarrow}$ if for each non-zero profile attribute of $\alpha^{\rightarrow}$ and $N^{\rightarrow}$, their bitwise 'XOR' operation results in all "1"s in the resultant vector.

Because the spatiotemporal relationships are straightforward, here we focus on profile bounding vector relationships between an authorization and a node. First, in case of $\supset_p$ relationship, observe that for every bit value of '0' of $\alpha^{\rightarrow}$, the corresponding bit value of $N^{\rightarrow}$ must be '0' because there must not exist any profile attribute value that only $N^{\rightarrow}$ includes but $\alpha^{\rightarrow}$ does not. Therefore, bitwise 'OR' operation would generate the same value with $\alpha^{\rightarrow}$. Also, in case of $\cap_p$ relationship, we need to see if there exists any common profile attribute value between α and $N^{\rightarrow}$. Therefore, if bitwise 'AND' operation results in a non-zero profile vector, we know that there exists common value set. Finally, in case of $\otimes_p$ relationship, we know α and $N^{\rightarrow}$ should not share

[5] A non-zero profile attribute vector refers to a binary vector that includes the value "1" in at least one bit.

Table 2. Bitwise Operation Results

$\alpha^{\rightarrow}$	$N^{\rightarrow}$	AND	OR	XOR	Relationship
110	011	010	111	101	$\alpha^{\rightarrow} \cap_p N^{\rightarrow}$
110	010	010	110	100	$\alpha^{\rightarrow} \cap_p N^{\rightarrow}$, $\alpha^{\rightarrow} \supset_p N^{\rightarrow}$
110	001	000	111	111	$\alpha^{\rightarrow} \otimes_p N^{\rightarrow}$

any profile attribute value that is common to each other. The bitwise 'XOR' operation is used for checking this condition, and the result of 'XOR' must include all '1's in the resultant $N^{\rightarrow}$.

Suppose $\alpha^{\rightarrow} = \langle **, 110, ** \rangle$, which implies that the authorization α is given to the users with salary $<$ \$62,000. Observe that because α evaluates the profile attributes 'Salary' only, we do not evaluate other profile attributes such as 'Department' or 'Home Town'. Therefore, as long as a user's salary is less than \$62,000, she meets the profile conditions of α. Considering the same PV_1^B, PV_2^B, PV_3^B in the previous section, suppose $N_1^{\rightarrow} = PV_1^B$, $N_2^{\rightarrow} = PV_2^B$, and $N_3^{\rightarrow} = PV_3^B$. We know that $N_1^{\rightarrow}$ and $N_2^{\rightarrow}$ include a user within this salary range while $N_3^{\rightarrow}$ does not include any user within the specified salary range. Also, in case of $N_2^{\rightarrow}$, all the value ranges of profile attributes for $N_2^{\rightarrow}$ are also included in $\alpha^{\rightarrow}$. Table 2 shows the results of bitwise AND, OR, XOR operations between $\alpha_1^{\rightarrow}$ and $N_1^{\rightarrow}$, $N_2^{\rightarrow}$, and $N_3^{\rightarrow}$ with their profile bounding vector relationships.

Authorizations Overlaying: The overlaying strategy traverses the S^{STP}-tree from the root node to leaf level by recursively comparing both the spatiotemporal extents and PV^Bs of the overlaying authorization and each node in the traversal path. Let us denote the spatiotemporal extent of a node N as $N^{\square}$, and PV^B as $N^{\rightarrow}$. All the possible scenarios for this comparison are as follows:

- **Case 1:** If $(\alpha^{\square} \supset_{st} N^{\square}) \wedge (\alpha^{\rightarrow} \supset_p N^{\rightarrow})$ is true, we stop traversing and overlay α on N. This overlaying strategy has several benefits. First of all, we overlay the authorizations on the first node encountered on the traversal path that totally encloses the spatiotemporal region and PV^B. As a result, authorizations are overlaid as high up as possible in the tree [4]. Because user access request evaluates also from the root node to the leaf level, authorizations that have been issued for the subject of the access request would be encountered as early as possible. Due to our overlaying strategy, existence of a relevant authorization in the traversal path for a subject of the access request means that all the moving objects stored at the subtree rooted at the node are already authorized. Therefore, we do not need to evaluate authorizations for the access evaluation process for the subtree. Observe that after overlaying an authorization on a node, it is not necessary to overlay the same authorization on any of its descendants.
- **Case 2:** Else if $(\alpha^{\square} \otimes_{st} N^{\square}) \vee (\alpha^{\rightarrow} \otimes_p N^{\rightarrow})$ is true, we stop the overlaying process. This is because, if subjects of α do not have a privilege to $N^{\square}$ or $N^{\rightarrow}$, α is not applicable to moving objects stored at the subtree rooted at N. Also, because $N^{\square}$ and $N^{\rightarrow}$ includes all the spatiotemporal extents and PV^Bs of all of N's descendants, there is no reason to traverse further to the leaf level.

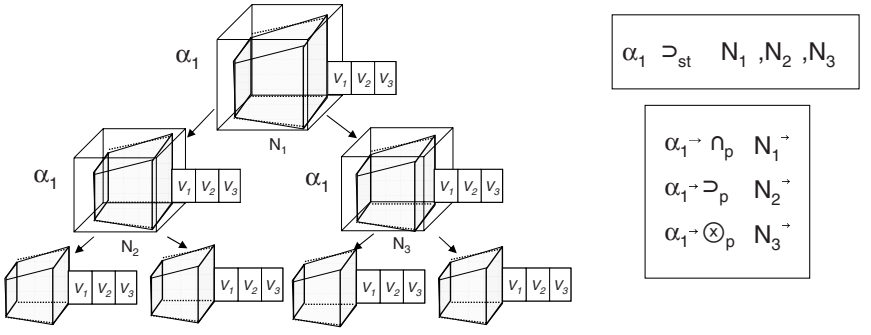

Fig. 4. Authorization Overlaying Process in S^{STP}-tree

– **Case 3:** Else if $(\alpha^{\square} \cap_{st} N^{\square}) \vee (\alpha^{\rightarrow} \cap_{p} N^{\rightarrow})$ is true, the overlaying strategy is different depending on the level of N.
  - If N is a non-leaf node, we traverse to each of N's children node C, and the same comparison between α and C is processed. This is because there may exist a descendent node whose spatiotemporal extent and PV^B is enclosed by that of α.
  - If N is a leaf node, we overlay α on N. This is because at least one of the moving objects stored in N comply with the spatiotemporal and profile specification of α. Therefore, in order not to discard any relevant authorization, we need to overlay α on N.

Figure 4 presents the overlaying process in the S^{STP}-tree. It shows that a node N_1 is a root node of the tree, and N_2, N_3 are the children nodes of N_1. Consider an authorization α_1 to be overlaid on the S^{STP}-tree. α_1 cannot be overlaid on the node N_1 since $\alpha_1^{\rightarrow} \cap_p N_1^{\rightarrow}$, which belongs to the case 3 above. Therefore, we need to traverse down to N_1's children nodes N_2 and N_3. The first traversal path is to N_2, and α_1 can actually be overlaid on N_2 because $\alpha_1^{\square} \supset_{st} N_2^{\square}$ and $\alpha_1^{\rightarrow} \supset_p N_2^{\rightarrow}$, which is case 1. Another traversal path to N_3 is stopped because $\alpha_1^{\rightarrow} \otimes_p N_3^{\rightarrow}$, which belongs to case 2.

5 User Access Request Evaluation

In this section, we present the details of user access request evaluation. Typically, a user request is of the form of requesting objects in the area of interest that satisfy a certain profile criteria. For example, a merchant is interested in sending promotion deals to mobile customers who are near a mall and whose salary is greater than $52,000. However, such promotion deals should be reached to only to the customers who are willing to reveal their salary information to that merchant (specified in the authorization) to receive the promotion deal.

A user request is denoted as $U = \langle s, \square, V, m \rangle$ where s is the subject requesting access, $\square$ is the spatiotemporal extent that the subject is interested in, V is interested profile vector, and m the access mode. We denote $U.s$, $U^{\square}$, $U^{\rightarrow}$, and $U.m$ to denote the

subject, the spatiotemporal extent, the profile vector, and the access mode of the user access request U, respectively. For example, if a merchant A wants to locate mobile customers who are 10 miles from the shopping mall and whose salary is greater than \$52,000, the user request would be $U = \langle$merchant_A, circle((50,60),10),$\langle 011 \rangle$, locate$\rangle$.

Algorithm 1. UserAccessRequestEvaluation

```
1: Input: node N, User Access Request U, Boolean authorized
2: Output: a set of authorized moving objects resultSet
3: if U^□ ⊗_{x,y,t} N^□ OR CheckPV(U^→, N^→) = disjoint then
4:    return NIL
5: end if
6: if There exists overlaid authorizations in N then
7:    Λ(N) ← CheckUserIDAuth(U, N)
8: end if
9: resultSet ← NIL
10: if authorized = false AND Λ(N) ≠ ∅ AND N is a non-leaf node then
11:    if Λ(N) ≠ NIL AND U^□ ∩_{S,T} N^□ AND CheckPV(U^→, N^→) = disjoint then
12:       authorized ← true
13:    end if
14: else if authorized = false AND Λ(N) ≠ ∅ AND N is a leaf node then
15:    for each α in Λ(N) do
16:       if N^□ ∩ U^□ AND CheckPV(U^→, N^→) = overlap) then
17:          resultSet ← resultSet∪ evaluate(α, U, N)
18:       end if
19:    end for
20:    return resultSet
21: end if
22: if authorized = true AND N is a leaf node then
23:    return evaluate(U, N)
24: end if
25: for each child c in N do
26:    MUserAccessRequestEvaluation(c, U, authorized)
27: end for
```

Algorithm 1 discusses the details of user access request processing. The initial function call is UserAccessRequestEvaluation($R, U, false$) where R is a root node of S^{STP}-tree. The evaluation process starts with the root node by comparing the spatiotemporal extents and the profile vectors of the user request and each node N involved in the top-down traversal. At the same time, the evaluation process searches for the relevant authorizations.

Given a user request U, we say an authorization α as relevant to U if the set of subjects evaluated by $\alpha.se$ includes $U.s$ and $U.m \prec_m \alpha.m$ for $U.s$ overlaid on the node N. We denote the relevant authorizations at a node N on the tree as $\Lambda(N) = \{\alpha \in$ overlaid authorizations on $N \mid U.s \in \alpha.se, U.m \prec_m \alpha.m \}$. The comparison among U, N and $\Lambda(N)$ during the traversal results in the following cases.

Case 1: $(U^{\square} \otimes_{st} N^{\square}) \vee (U^{\rightarrow} \otimes_{p} N^{\rightarrow})$ is true: The disjoint relationship implies that all the moving objects stored at the subtree rooted at N are not within the spatiotemporal region or do not meet the profile condition for the user request U. Regardless of the existence of relevant authorizations for U at N, the moving objects stored at the subtree rooted at N are not within the user's interests. Therefore, the traversal stops regardless of the existence of overlaid authorizations.

Case 2: $(\Lambda(N) \neq \emptyset) \wedge ((U^{\square} \cap_{st} N^{\square}) \vee (U^{\rightarrow} \cap_{p} N^{\rightarrow}))$ is true: If N is a non-leaf node, although all the moving objects stored at the subtree rooted at N are authorized, the user wants to retrieve a subset of moving objects whose locations are within $U^{\square}$ and whose profiles are enclosed by $U^{\rightarrow}$. Therefore, for the subtree rooted at N, we retrieve moving objects whose location overlaps with $U^{\square}$ and whose profile condition overlaps with $U^{\rightarrow}$. We do not need to evaluate authorizations during the traversal because the subtree rooted at N is already authorized by $\Lambda(N)$.

If N is a leaf node, because we overlay authorizations on a leaf-node in an enclosing case as well as overlapping case, not all of the moving objects in N are authorized. Thus, for all $\alpha \in \Lambda(N)$, return the moving objects that are located within $\alpha^{\square} \cap_{st} U^{\square}$ and whose profiles are overlapped with $\alpha^{\rightarrow} \cap_{p} U^{\rightarrow}$.

Case 3: $(\Lambda(N) = \emptyset) \wedge ((U^{\square} \cap_{st} N^{\square}) \vee (U^{\rightarrow} \cap_{p} N^{\rightarrow}))$ is true: If N is a non-leaf node, access control decision cannot be made because there is a possibility that a relevant authorization may be overlaid on a descendent node of N. Thus, evaluation process repeats for all the children nodes of N. If N is a leaf node, we reject the access request because there exists no relevant authorization for U during the traversal.

Case 4: $(\Lambda(N) \neq \emptyset) \wedge ((\mathrm{U}^{\square} \supset_{st} N^{\square}) \wedge (U^{\rightarrow} \supset_{p} N^{\rightarrow}))$ is true: There exists at least one relevant authorization for U is overlaid on N. If N is a non-leaf node, because the spatiotemporal extents and profiles stored at the subtree rooted at N are authorized, all the moving objects stored at leaf nodes of the subtree rooted at N are allowed to be accessed by $U.s$. Therefore, there is no need to evaluate authorizations on the subtree rooted at N. In addition, spatiotemporal and profile vector comparisons would not be required either because all the moving objects stored at the subtree rooted at N are within the user's interests. If N is a leaf node, some of the moving objects in N may not meet the conditions set by U. Thus, for all $\alpha \in \Lambda(N)$, the algorithm would return all the moving objects that are located within $\alpha^{\square}$ and whose profiles are overlapping with those of $\alpha^{\rightarrow}$.

Case 5: $\Lambda(N) = \emptyset \wedge ((U^{\square} \supset_{st} N^{\square}) \wedge (U^{\rightarrow} \supset_{p} N^{\rightarrow}))$ is true: Although all the moving objects stored at the subtree rooted at non-leaf node N meet the spatiotemporal and profile conditions of U, access control decision cannot be made because there is a possibility that a relevant authorization may be overlaid on a descendent node of N. Thus, evaluation process repeats for all the children nodes of N. If N is a leaf node, we reject the access request because there exists no relevant authorization for U.

Table 3. Number of Disk Access

S^{STP}-Tree		Separate Index Structures	
If $m < M - k/s$	Else	Index for moving objects	Index for profiles
$\Omega(\log_m N)$	$\Omega(\log_{M-k/s} N)$	$\Omega(\log_m N)$	$\Omega(\log_m N)$

Note that, in algorithm 1, the operation CheckUserIDAuth() returns relevant authorizations for U among the overlaid authorizations in the node N. CheckPV(A, B) returns $overlap$ (if $A \cup_p B$), $disjoint$ (if $A \otimes_p B$), and $enclose$ (if $A \supset_p B$). The overloading function evaluate() returns the moving objects whose location and profile conditions meet the user request, and which are stored in the leaf node N.

User Access Request Performance Analysis: We present an informal analysis of the complexity of user request evaluation by comparing the performance between the proposed S^{STP}-tree and the where there are two separate index structures (one for moving objects and another for profile). For the discussion, we do not consider authorizations because the overlaying procedure does not change the structure of the tree. Overlaying simply stores the relevant authorizations on the nodes of the tree, which does not incur any changes on the structure of the tree.

For the analysis, let us suppose the following:

- N is the number of moving objects: That is, there are N number of location information and N of profile vectors.
- The number of children (or data) that each non-leaf (or leaf) node includes is between m and M where $2 \leq$ m $\leq M/2$: this implies that the height of the tree is bounded by [$\log_M N$, $\log_m N$]
- k is the size of PV^B in bytes
- b is the disk page size in bytes
- s is the size of location information and profile vector (in bytes)

If a tree does not store a PV^B in a node, $M = b/s$ because M is the maximum number of children node or data that can be stored in each disk page without considering the PV^B. However, in case of S^{STP}-Tree, k bytes are reserved to store a PV^B for each node. Thus, the maximum number of data (children) for each disk page is

$$(b - k)/s = b/s - k/s \tag{1}$$
$$= M - k/s \tag{2}$$

Thus, the height of S^{STP}-tree is
- If $m < M - k/s$, the height = [$\log_m N$, $\log_{M-k/s} N$].
- Else, the height = [$\log_{M-k/s} N$, $\log_m N$]

The number of disk accesses for user request is summarized in 3. If $m < M - k/s$, it is obvious that the S^{STP}-tree shows better performance (fewer disk accesses) because in this case, the proposed tree would generate the exactly same structure of tree as

that from the separate index for moving objects. Thus, separate indexes would need to access the number of nodes from the profile index additionally than the S^{STP}-tree.

6 Conclusions

Recently, unified index structures, ^{s}TPR-tree and S^{PPF}-tree, have been proposed to organize both moving objects and authorizations specified over them. However, both approaches are not capable of evaluating security policies that include profiles because the index structure can support only spatiotemporal regions of security policies. In this paper, we have proposed an index structure, called the S^{STP}-tree, which enables one to enforce and evaluate the security policies that include profiles of moving objects. We provide an informal analysis to show that the proposed structure is more efficient than maintaining indexes independently for moving objects and authorizations, and profiles. Currently, we are in the process of implementing the proposed tree to experimentally validate the gain in performance with respect to the independent cases.

References

1. Youssef, M., Adam, N.R., Atluri, V.: Preserving Mobile Customer Privacy: An Access Control System for Moving Objects and Customer Information. In: 6th International Conference on Mobile Data Management. LNCS. Springer, Heidelberg (2005)
2. Wolfson, O., Xu, B., Chamberlain, S., Jiang, L.: Moving objects databases: Issues and solutions. In: Rafanelli, M., Jarke, M. (eds.) 10th International Conference on Scientic and Statistical Database Management, Proceedings, Capri, Italy, July 1-3, 1998, pp. 111–122. IEEE Computer Society Press, Los Alamitos (1998)
3. Moreira, J., Ribeiro, C., Abdessalem, T.: Query operations for moving objects database systems. In: Proceedings of the eighth ACM international symposium on Advances in geographic information systems, pp. 108–114. ACM Press, New York (2000)
4. Atluri, V., Guo, Q.: Unied index for mobile object data and authorizations. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 80–97. Springer, Heidelberg (2005)
5. Atluri, V., Shin, H.: Efficient enforcement of security polices based on the tracking of mobile users. In: DBSec, pp. 237–251 (2006)
6. Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the positions of continuously moving objects. In: SIGMOD Conference, pp. 331–342 (2000)
7. Atluri, V., Adam, N.R., Youssef, M.: Towards a unied index scheme for mobile data and customer proles in a location-based service environment. In: Workshop on Next Generation Geospatial Information (NG2I'03) (2003)
8. Atluri, V., Chun, S.A.: An authorization model for geospatial data. IEEE Trans. Dependable Sec. Comput. 1(4), 238–254 (2004)
9. Gazetteer, U.S.: `http://www.census.gov/cgi-bin/gazetteer`
10. Oracle corporation data sheet - oracle spatial option and oracle locator: Location features in oracle database 10g. Technical report, Oracle (2004)`http://www.oracle.com/`
11. Atluri, V.: Mobile commerce. In: In The Handbook of Computer Networks, Volume III Distributed Networks, Network Planning, Control, Management and Applications, Part 3: Computer Network Popular Applications, John Wiley & Sons Inc., West Sussex, England (2007) (page to appear)

12. Mobile Commerce (M-Commerce) & Micropayment Strategies. Technical report, Juniper Research (2004) http://www.juniperresearch.com/
13. Venkatesh, V., Ramesh, V., Massey, A.P.: Understanding usability in mobile commerce. Commun. ACM 46(12), 53–56 (2003)
14. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: International Conference on Machine Learning, pp. 194–202 (1995)