

Preserving Mobile Customer Privacy: An Access Control System for Moving Objects and Customer Profiles

Mahmoud Youssef

Vijayalakshmi Atluri*

Nabil R. Adam

MSIS Department and Center for Information Management, Integration, and Connectivity (CIMIC)

Rutgers University

180 University Avenue, Newark, NJ 07102, USA

+1(973) 353 1014

{youssefm, atluri, adam}@cimic.rutgers.edu

ABSTRACT

A key challenge for Mobile services is to offer personalized contents while preserving the privacy of customers. In mobile applications, location information is modeled as moving objects. Providing proper protection to customer information can be achieved by an access control system. However, providing such system is a challenging task due to: 1) the spatio-temporal nature of the constraints as well as the location information, and the interaction among them; 2) the complexity of resolving spatio-temporal and granularity conflicts; and 3) the required scalability and efficiency. In this paper, we present a solution that includes an access control model for moving objects and customer profiles. We also present a mechanism that enforces the spatio-temporal policies. The mechanism consists of three components: a text encoder, a spatio-temporal module that computes interactions between moving objects and spatio-temporal constraints, and a new data structure referred to as the Adaptive Search Multi-way trie (ASM-trie). We present the insertion and search algorithms of the ASM-trie and an evaluation study that shows the positive impact of the ASM-trie on the search efficiency.

Categories and Subject Descriptors

H.3.1 [Information Systems]: Information Storage and Retrieval – *Content Analysis and Indexing, Indexing Methods.*

D.4.6 [Software]: Operating Systems – *Security and Protection, Access Control.*

K.4.1 [Computer Milieux]: Computers and Society – *Public Policy Issues, Privacy.*

General Terms

Algorithms, Performance, Security.

Keywords

Moving Objects, Customer Profiles, Access Control, Privacy, Mobile.

MDM 2005 05 Ayia Napa Cyprus

(c) 2005 ACM 1-59593-041-8/05/05....\$5.00

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1. INTRODUCTION

Due to advancements in such technologies as wireless networking and the Global Positioning System (GPS), it is possible to track different entities such as vehicles, containers, and individuals. Location information is typically maintained by a Location Service (LS) that supports other mobile services (M-Services) and applications. One of the major applications expected to benefit from the existence of LS is the mobile-targeted marketing, or location-based advertising. In that application, merchants can increase their sales by sending customers highly personalized offers that are based on two types of information: customer attributes (profile) and customer location. While such applications can offer customers the convenience of personalized location-based services, the improper use of customers' information can present a serious threat to their privacy.

Personalization is achieved by selecting customers who satisfy certain criteria. In mobile applications, a merchant applies the profile and location criteria by querying the corresponding databases. Each merchant maintains her own copy of the profiles database locally. On the other hand, the LS aggregates location data from different sources such as mobile devices equipped with GPS and the wireless infrastructure. Since customers can be continuously moving, a large number of customers updating their location can create an update problem to traditional databases. The Moving Objects Database (MOD) [20, 22] has been proposed to manage location data.

The lack of privacy protection in mobile services may adversely impact a promising market that is just taking off. In that market, location-based services are among the most promising. Nevertheless, the public concerns about privacy have to be addressed for these services to realize their benefits for customers and businesses.

In addition to the public, lawmakers, the industry, and academia have recognized the importance of addressing location privacy. The Internet Engineering Task Force (IETF) has approved a working group on geographic location privacy (GeoPriv). The lawmakers in the USA, Canada, and Europe have passed several laws that mandate obtaining customer consent before using her location information [9]. Obtaining such consent is typically implied by the opt-in process.

*The work of V. Atluri is supported in part by the National Science Foundation under grant IIS-0242415

Chellappa and Sin [6] found that most customers do not opt-in to personalized online services unless they trust the vendor for their profile information. They also found evidence that customers are willing to trade-off their information with a trusted vendor for convenience. That evidence was even found among the most privacy-concerned customers. It is, however, reasonable to expect that mobile customers would be more concerned with their privacy especially when their location information can be linked to their profiles.

Privacy can be defined as the personal control over private information. Such control is usually assured to the customer through a set of policies that are enforced by some technical means. Since the private information is stored in databases, an access control system can precisely enforce these policies. The scope of this paper is limited to the customer privacy policy. Other policies, such as administrative policies, are not covered.

1.1 Problem Statement

In the mobile application and services environment, there are several players each playing a different role: the LS, the information requester (e.g. the merchant or a marketing intermediary), the information provider (e.g. the wireless network), and the customers. An effective privacy policy has to meet, at least, two goals: preventing unauthorized sharing of customer information among information requesters; and preventing misuse of permitted access to this information.

One of the major sources of the spam problem is the frequent sharing and aggregation of customer information among authorized and unauthorized marketers. While personalization and customer convenience require making decisions based on this information, privacy protection requires ensuring that it is not disclosed improperly. In addition, if the access policies are based only on the merchant's identity, the customer may receive offers at times and locations that do not match their preferences. Therefore, such preferences have to be part of the privacy policy. That is, the customer must be able to specify who has access to her information at what times and locations. As such, the access rules that detail the policy have to include spatio-temporal constraints.

Enforcing spatio-temporal constraints for mobile customers requires two capabilities: 1) translation between geospatial coordinates, as expressed in the MOD, and civil names, as expressed in the constraints; and 2) addressing the impact of customer motion. The typical location query includes a spatial window and a time interval. Since the query may overlap with more than one location and time units (as defined in the constraints), then multiple access rules for the same customer may need to be evaluated and the conflict among their results to be resolved. We refer to this as *spatio-temporal conflict*.

Another important requirement is the support of a user-friendly approach to defining policies. Instead of specifying access rules to individual merchants at every location and time, the customer should be able to define rules at different levels of granularity. Yet, granular representation may increase the size of the access control system; and it requires resolving conflict among the results of the different access rules defined for the same entity but at different granularities. We refer to this as *granularity conflict*.

Finally, the access control system has to be scalable and efficient. It has to accommodate the growth in the number of merchants and customers [10]. In addition, it should not adversely impact the overall response time of the query processing [11].

Satisfying the above requirements, makes the task of designing an access control system a challenging task due to several reasons: 1) the spatio-temporal nature of the constraints as well as the location information, and the interaction among them; 2) the complexity of resolving spatio-temporal and granularity conflicts; and 3) the required scalability and efficiency.

The rest of the paper is organized as follows: In Section 2, we present the proposed solution. We then follow by an evaluation study in Section 3, and the related work in Section 4. In Section 5, we present a summary and future work.

2. THE PROPOSED APPROACH

2.1 One Trusted Party

When a customer trusts too many merchants for her profile, it becomes difficult to protect it or hold the merchants accountable if they share it. A remarkable improvement can be gained by limiting the number of parties to be trusted to one trusted third party. Considering that most of the proposed architectures (e.g. RFC 3936) are based on the existence of a LS and that the customer trusts it for their location information, it stands to reason then that the LS assumes the role of the trusted third party. Hence, access control would be enforced by the LS. This choice is plausible for two reasons: 1) enforcing spatio-temporal policies requires spatio-temporal processing which the LS is normally capable of; 2) a LS is seen to be implemented as a globally distributed service [11, 16, 17] which reduces the system susceptibility to the two major vulnerabilities: being a single point of failure, and being attractive to hacking attacks.

2.2 Controlling Information Flow

Merchants may maliciously collaborate and aggregate their views of customer data [10]. Therefore, it is necessary that the information reported to the merchant cannot be linked directly or indirectly to any specific customer. However, it is usually desired to track customer's purchase history. In addition, it may be necessary to provide such information for the purpose of service charges. We address this problem by requiring information to move in one direction from the merchant to the LS to the wireless network and finally to the customers. The results reported to the merchant are dynamic pseudonyms of the customers who received the offer. These pseudonyms can be created using one-way hash function. Thus, accounting information is reported to the merchant but he cannot gather or share customer information. The steps are depicted in Figure 1 and listed below. These steps are tailored for a *push* scenario. In a *pull* scenario, non-expiring offers are stored at the location service until the customer fetches them.

- 1- Merchants send information related to a specific offer along with the query to the LS;
- 2- The LS runs the query producing a list of IDs;
- 3- The LS enforces access control which filters the IDs;
- 4- The filtered IDs are then forwarded with the offer to the wireless networks to deliver it to the customers;
- 5- The wireless networks send the offers to the customer devices; then

6- Upon success reports from the networks, the LS sends pseudonyms to the merchant.

For the purpose of service charges, the pseudonyms are sufficient. However, for tracking consumer’s purchase history, the consumer has to complete the transaction using that pseudonym. This will require anonymous payment service which has been discussed in the literature (e.g., [2]) and probably anonymous shipment service also if the transaction involves physical goods.

2.3 The Access Control Model

In early work on access control all authorizations were positive and the authorization consisted of a triple <subject, object, access modes>. Such an authorization asserts what access modes the subject can exercise on the object. In recent work, authorizations are extended by a flag that indicates whether the access modes are granted or denied [3]. In advanced models, an access rule can also have temporal, spatial, and spatio-temporal *constraints*. These constraints define the context where the authorizations are applicable, and they usually employ special operators to express relations in that context. For instance, temporal constraints may include such operators as *whenever*, *aslongas*, *unless*, *whenevernot*. Similarly, spatial constraints may utilize operators such as *within-distance*, *contains*, and *intersects*.

2.3.1 Formulation of Access Rules

We propose a discretionary access control model where the access rules are set by the customer. The authorization components are mapped as follows: the merchants (information requesters) are mapped to Subjects; customer information (profile and/or location) is mapped to Objects; and the only Access Mode is “read”. Since there is only one access mode, it can be removed from the authorization. Therefore, the access rule in the model is composed of an authorization triple, and spatio-temporal constraint:

$$(s, o, +/-), <stc>$$

Where

- s is the subject described at some level of granularity;
- o is the object which consists of a customer ID and a subset of $\{l/p\}$ where l is the location information and p is the customer profile;
- $+/-$ is a flag; and
- stc is a civil location and a time interval.

The proposed model employs both positive and negative authorizations as follows: each customer starts with a *global denial rule* that assigns negative access to all merchants on all objects at all locations and times. Later, the customer can change the flag of this rule or add other rules that override it. In general, a customer can partially override any access rule by specifying a more specific (refined) rule. The notion of relative specificity among rules is detailed later.

2.3.2 Model Components Representation

In the proposed model, each component, except the customer ID and the flag, is represented as a hierarchy. These hierarchies hold several properties:

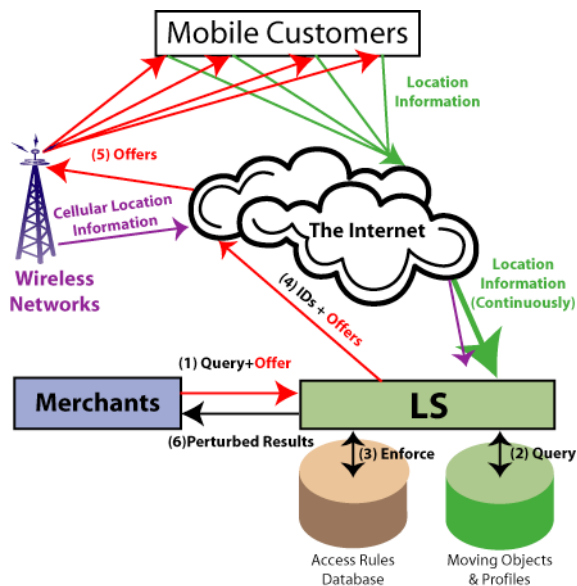


Figure 1. Information flow in the system

- 1- In every level in a hierarchy, the nodes are exact decomposition of their parents. That is, the parent is the union of the children and the children are in “disjoint” relation. Due to that decomposition, the root always represents “All Members” and the leaves are the members at their most specific representation. For instance, in the subject hierarchy (referred to as the Industry hierarchy, see Figure 2) the root is “All Industries”, the leaves are the individual merchants, and the intermediate nodes are the parent industries of the merchants.
- 2- The number of levels in a hierarchy indicates the levels of granularity, for instance the location hierarchy can have four levels (Country, Region, State, and County).
- 3- The nodes along the path from a leaf to the root represent a member at its different levels of granularity. For instance, as depicted in Figure2, the path including Hotels, Lodging, Personal Services, and All Industries can be seen as other granularities of the Hilton.

We arrange the hierarchies in an order in which the first hierarchy after the ID, also the simplest, is the object hierarchy which is composed of two levels: the root is $\{l/p\}$ and the leaves are $\{l\}$ and $\{p\}$. After the object hierarchy comes the subject, the location, and then the time hierarchy. Semantically, the order among the object, subject, location, and time hierarchies implies *precedence*, however, that precedence does not affect the model behavior as long as the same order followed in the specification is also followed in the evaluation of the access policy.

At any point, the system state includes multiple partial instantiations of these hierarchies. That is, in the system state, each instance consists of those nodes which are included in existing rules. Therefore, each customer always has an instance of the object hierarchy. In that instance, each of the nodes has an instance of the subject hierarchy and so forth for the other components. The nodes in the time hierarchy point to flags (Figure 3). The instances belonging to an individual customer can be seen as a tree; hence, the entire model can be seen as a forest.

2.3.3 Resolving Conflicts

As discussed in the problem statement, two types of conflicts may arise: spatio-temporal, and granularity. We use two different strategies in resolving conflicts. For resolving spatio-temporal conflicts, we apply the *denial precedes grants* rule. This makes the model more biased to customer privacy. For granularity conflict, we follow the *inheritance with overriding* strategy.

In the later strategy, inheritance is applied when no rules are specified for a node in the hierarchy, i.e. the node does not exist in the instance. The node is assumed to virtually exist and “inherits” the constraints from the first existing ancestor in the instance. For example, if a customer defined an access rule on Hotels, but not on the Hilton, then the Hilton inherits the constraints on Hotels. Among existing nodes, overriding is applied. Therefore, more specific access rules “override” less specific ones. In Figure 2, if two identical access rules, except for the subject, are defined on Personal Services and Lodging, then the flag of the Lodging rule will override the flag of the Personal Services rule.

The relative specificity among rules is determined as follows: consider two rules R1 and R2 that have the same customer ID, then R1 is more specific than R2 in all of the following cases regardless of the rest of the rule:

1. R1 has a more specific object than R2; i.e. R2 has $\{l+p\}$ and R1 has l or p .
2. R1 and R2 have the same object AND R1 has a more specific subject;
3. R1 and R2 have the same object and subject AND R1 has a more specific location;
4. R1 and R2 have the same object, subject, and location AND R1 has a more specific time.

The proposed modeling provides several advantages:

1. The search process is more efficient. Overriding results suggests having search starts from most-specific representation. We exploit that in resolving granularity conflict by adaptively searching for the most specific rule that matches some search key. Figure 4 depicts an example of an adaptive evaluation of a search key. In the figure, original search path is in the most specific representation. Upon failure to find a node, the search retreats to its parent and tries to continue from there.
2. By using both positive and negative flags, the instances of the hierarchies are partial instantiation, therefore, the system state is kept small. This improves scalability and efficiency.
3. Component representation is granular. This streamlines the user interface and provides support for aggregate queries.

2.4 Evaluation of Access Control

Customer motion has the greatest impact on evaluation of access control. The continuous movement of a large number of customers makes the pre-computing of access rules an extremely inefficient operation. The data representation for the MOD has been addressed in several proposals. Nevertheless, due to its simplicity and practicality, the Moving Objects Spatio-Temporal

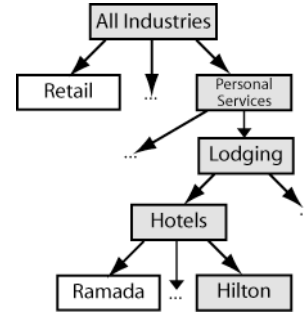


Figure 2. A portion of the industry hierarchy (the Subject hierarchy)

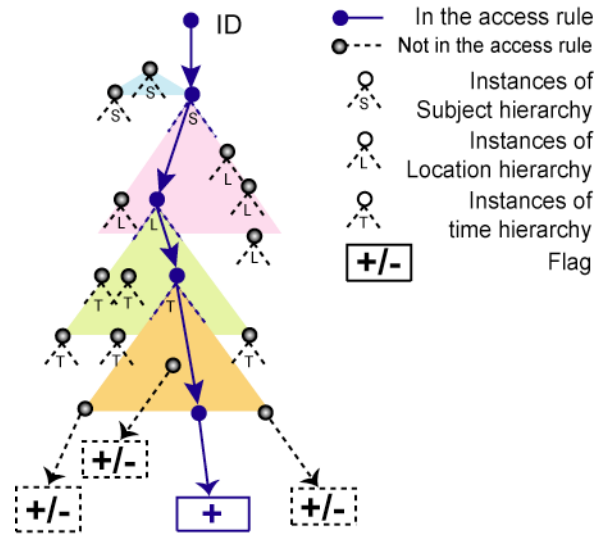


Figure 3. A customer tree showing instantiations of the different hierarchies and an example of an access rule

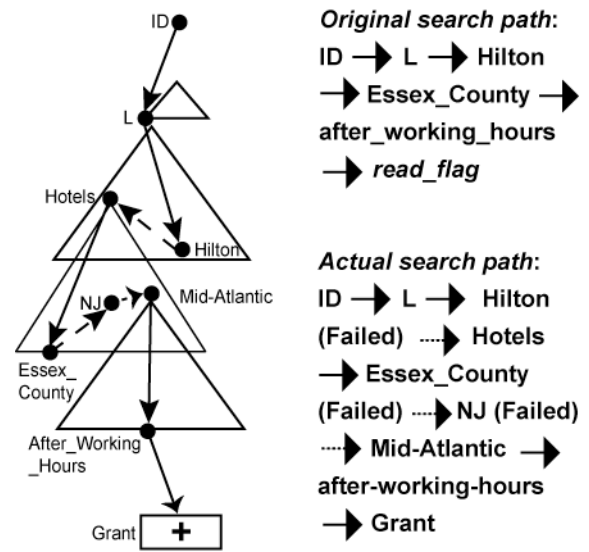


Figure 4. Adaptive search

model (MOST) [20] has been adopted by most of the indexing techniques in the literature (e.g. [7, 15, 19, 21]). The authors of the MOST have also proposed the Future Temporal Logic (FTL) and an associated query language. The MOST has also been extended to address issues such as uncertainty and update policies [22].

The MOST supports several types of queries that can usually be reduced to a *spatial window* with a *time interval* query [21]. It also supports *predictive queries* which include time intervals about the near future.

In the MOST, location information is treated as a dynamic attribute and is represented as a linear function of time. This linearity assumption has been investigated and found to be reasonable especially for applications where the objects move on a predefined network of roads [14, 15]. Moreover, it is compatible with the Universal Mobile Telecommunication Service (UMTS); the third generation wireless networks [23].

Evaluation of access control involves searching for some keys in the access rules database and retrieving the matching flags. In the proposed model, the search keys consist of an object, a subject, a location, and a time interval. Each customer in the query results can generate a set of search keys based on the locations and times that the query intersects. While granular representation is also another source of multiple search keys, we apply adaptive search to eliminate the need for composing those keys. In the following discussion, we use the term “spatio-temporal window” to refer to a combination of a civil location and a time interval at the leaf levels. We also use “customer” in the following discussion to refer to a customer who satisfies the query. The following scenario outlines the evaluation procedure:

Consider a query submitted by the Hilton for all customers in Essex County (which is part of the state of NJ, USA) for a time interval 4:45 to 5:15 pm (which intersects with both time leaves “During Working Hours” and “After Working Hours”). Assume that all customers who satisfy the query will be moving inside Essex County during that time interval.

The evaluation proceeds as follows:

- 1- For each customer and for each spatio-temporal window that the customer passes through, a search key is created. In this case, two spatio-temporal windows are possible: (Essex County, During Working Hours) and (Essex County, After Working Hours). The search key includes the Object $\{l\}$, the merchants ID as Subject, and a spatio-temporal window as a constraint.
- 2- For each of the created keys, an adaptive search operation is performed and a flag is retrieved (see Figure 4).
- 3- The flags that belong to the same customer are combined using the denial precedes grants rule.

We now formalize this procedure, show the computation for finding the spatio-temporal windows, and relax the single-location assumption set above.

Let Q be a set of queries $\{q_i\}$ where query q_i is represented by a spatial window $[(x_{i1}, y_{i1}), (x_{i2}, y_{i2})]$, a time interval $[t_{i1}: t_{i2}]$ and a requested information $\subseteq \{l_p\}$. Also let,

C be the set of all customers who satisfy the query $\{c_j\}$;

K_i be the set of search keys for q_i for the customers in C ;

$K_{ij} \subseteq K_i$ be the subset of search keys $\{k_{ijm}\}$ belonging to customer c_j for the query q_i ;

P_i be the set of flags for query q_i ;

$P_{ij} \subseteq P_i$ be the subset of flags $\{p_{ijm}\}$ belonging to customer c_j from K_{ij} ;

L be the set of all leaf locations $\{l_n\}$;

V be the set of customer speeds $\{v_j\}$, where v_{jx} and v_{jy} are the speed components of customer c_j in the direction x and y respectively [19]; and

F be the set of linear motion functions $\{f_j\}$ as defined by the MOST, that is, at any time t :

$$f_j(t) = \text{sqr}t((x_j(t))^2 + (y_j(t))^2),$$

$$x_j(t) = x_j(t_{ref}) + v_{jx}(t - t_{ref}), \text{ and}$$

$$y_j(t) = y_j(t_{ref}) + v_{jy}(t - t_{ref}). \text{ Where } t_{ref} \text{ is a reference time.}$$

To simplify the presentation, we include only the X-axis in Figure 5. The following is the procedure for obtaining the spatio-temporal windows:

- Obtain a set of temporal points T_i that represent the intersections between the set of motion lines $\{f_j\}$ and the query q_i . For each customer c_j , the motion line $\{f_j\}$ intersects the borders of the query q_i in two temporal points t_{ij1} and t_{ij2} that are calculated as follows:
 t_{ij1} is the smallest of
 $t_{ref} + v_{jx} * (x_{i1} - x_{ref})$ and $t_{ref} + v_{jy} * (y_{i1} - y_{ref})$
if $t_{ij1} = t_{i1}$, then t_{ij2} is the smallest of
 $t_{ref} + v_{jx} * (x_{i2} - x_{ref})$ and $t_{ref} + v_{jy} * (y_{i2} - y_{ref})$
else $t_{ij2} = t_{i2}$.
- For those two temporal points, we find the corresponding spatial points (x_{ij1}, y_{ij1}) and (x_{ij2}, y_{ij2}) as follows:
 $x_{ij1} = x_j(t_{ref}) + v_{jx}(t_{ij1} - t_{ref}),$
 $y_{ij1} = y_j(t_{ref}) + v_{jy}(t_{ij1} - t_{ref}),$
 $x_{ij2} = x_j(t_{ref}) + v_{jx}(t_{ij2} - t_{ref}),$
 $y_{ij2} = y_j(t_{ref}) + v_{jy}(t_{ij2} - t_{ref}),$
- Using digital maps, e.g., the USA Census Tiger, and the spatial operation *intersection*, we then find the leaf locations l_{ij1} and l_{ij2} . Thus, we can have one, two, or four spatio-temporal windows for each customer, with the exact number being dependent on whether t_{ij1} and t_{ij2} belong to the same time leaf, and whether (x_{ij1}, y_{ij1}) and (x_{ij2}, y_{ij2}) belong to the same location leaf.
- Each of the spatio-temporal windows is then appended to the customer ID, the requested information (the object), and the merchant ID (the subject) to create the set of search keys K_{ij} for customer c_j .
- Each key k_{ijm} is evaluated by the adaptive search process and a flag p_{ijm} is obtained. The set of flags for customer c_j is P_{ij} .
- The denial precedes grants rule is applied on this set of flags and a final flag is obtained for that customer.

Table 1: A Sample from location encoding

| Location | Code without padding | Code with padding |
|---------------|----------------------|-------------------|
| All USA | b | baaaa |
| New England | bb | bbaaa |
| Mid-Atlantic | bc | bcaaa |
| ... | | |
| New Jersey | bcd | bcdaa |
| New York | bce | bceaa |
| ... | | |
| Essex County | bcdbe | bcdbe |
| Hudson County | bcdbf | bcdbf |

2.5 The Enforcement Mechanism

Our strategy for achieving scalable and efficient enforcement is to turn the problem into a string search problem. The enforcement mechanism consists of three sub-modules: a spatio-temporal module, an encoder, and the ASM-trie. The spatio-temporal module generates the search keys. The encoder converts access rules and search keys to alphabetical strings, and the ASM-trie provides the adaptive search on the encoded strings. Finally, the main module in the mechanism applies the denial precedes grants rule.

2.5.1 The Spatio-temporal Module.

The role of this module is to carry out the computations for generating the search keys as detailed in Section 2.4. It also provides the spatial operation *intersection*. We developed the module on top of Oracle Spatial using C++ preprocessor.

2.5.2 The Encoder.

This module converts both access rules (excluding the flag) and search keys into alphabetical strings that are designed to support adaptive search. Each string consists of five substrings that represent the ID, an object, a subject, a location, and a time. These substrings are drawn from preprocessed tables where the substrings are unique in each table. The encoding method is designed to help the adaptive search in two ways: first, the path from the root to any node in a hierarchy is embedded in the substring of that node. For instance, the substring ‘bcdbf’ from Table 1 includes letter ‘b’ for the USA, letter ‘c’ for Mid-Atlantic, letter ‘d’ for New Jersey and the two letters ‘bf’ for Essex County. Second, all the substrings are made equal-length by adding a padding. Semantically, the padding letter, letter ‘a’, is equivalent to the parent (see Table 1). To avoid ambiguity, We did not use ‘a’ for encoding.

In creating the tables, the encoder creates a unique set of substrings for a hierarchy by assigning unique alphabetical letter to each of the nodes that share the same parent in that hierarchy. For instance, in the location hierarchy, the USA node has six children (regions). These regions are assigned the letters ‘b’ to ‘g’. However, some children may need more than one letter per node. This is the case when the number of children is more than 25. Since the ASM-trie is based on alphabetical radix and since one letter is used as padding, the trie node can accommodate up to 25 children (see Figure 7). This is often not sufficient, many industries such as the hotels industry has more than 25 children.

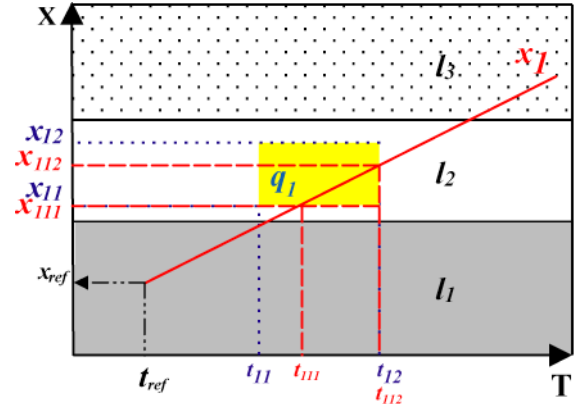


Figure 5. Impact of customer motion on search keys

Similarly, some states have more than 25 counties. We allocate multiple letters for such children. These multiple letters become a corresponding number of levels in the ASM-trie (See Figure 6). Since these multiple levels in the trie map to one level in the hierarchy, there is no one-to-one relation between the nodes in the hierarchy and the corresponding nodes in the trie.

The need for padding to indicate the parent path can be explained by the adaptive search process. Whenever the search fails and in order to adapt, it retreats to the immediate parent of the failed node and continues forward if the parent is found. Therefore, the parent path has to be easily identifiable. By designating the letter ‘a’ for padding, a failing search can look for letter ‘a’ and continues from there.

Adaptive search is performed on the hierarchy substrings but not the ID substring which has only one representation. While encoding, IDs are represented as uppercase letters. This notion is conveyed to the trie structure during the insertion of access rules by setting a Boolean variable, *adaptive-search*, to *false*. That variable is evaluated before the adaptive search can proceed.

As can be seen, the ASM-trie functionality depends largely on the semantics captured by the encoding.

2.5.3 The ASM-Trie

A trie is an M -ary tree whose nodes are arrays of pointers [8]. In the ASM-trie, the path from the root to a leaf is an access rule. With exception of the leaf node which represents the flag, all the nodes are similar and they include the following (see Figure 7): 28 pointers that represent the alphabet, the null character, and a *previous-letter* pointer for backward traversal. The node also includes the Boolean variable *adaptive-search*. When a node is created, all the pointers receive null values. A non-null pointer indicates that a letter has been inserted. That letter is implied by the *order* value of that pointer in the node; 0 = null, 1 = a, 2 = b, and so forth.

The ASM-trie relies on a class called *Access-Rule* to carry the access rules being inserted. It has two member variables: *ars*, the access rule string and a Boolean variable *f*, the flag. The class also has two accessor functions *getArs()* and *getF()*, and two mutator functions *updateArs(string)* and *updateF(bool)*. The next section details the insertion and search algorithms of the ASM-trie.

2.5.4 The ASM-Trie Algorithms

The basic difference between trie insertion and tree insertion is that in the trie, we insert a complete path from the root to a leaf rather than inserting one node. Part of the path may already exist. For example, inserting the string 'bcdbe', in Figure 6, will actually involve inserting the letters 'be' only since the other letters were inserted as part of the string 'bcdaa'. The insertion algorithm (see below) consists of a main function `Insert` that retrieves the string *ars* and the flag *f* from the access rule to be inserted. It then initializes four variables: the *position* variable to the beginning of the *ars* string, the *head* and *upLink* variables to the memory address of the root of the trie, and *isAdaptiveSearch* to false. Finally, it calls the recursive insert function `InsertR`.

`InsertR` basically extracts one letter at a time and insert it as a child node of the previous letter. The insertion of a letter is carried out by creating a new node and assigning its memory address to a pointer in the parent node. For instance inserting letter 'f', in Figure 6, involves reading the pointer at location 6 (the *order*). If the pointer value is null, a new node is created and its memory address is copied to the pointer in location 6, otherwise the letter already exists. In order to find the letter *order*, we subtract 64 from its ASCII code. If the value is more than 26 (i.e. lowercase), we subtract 32 and set the *isAdaptiveSearch* to true.

In addition to setting the letter pointer value, the insertion process assigns the previous node address, stored in the *upLink* pointer to the *previous-letter* pointer in the node. It also assigns the Boolean value in *isAdaptiveSearch* to the node variable *adaptive-search*.

When the *order* value of an extracted letter is null, i.e. end of string, the flag is inserted, if it does not exist and SUCCESS is returned. If the flag exists, FAILURE is returned to indicate that the access rule already exists.

The variable *head* guides the insertion process to the location where an extracted character should be inserted by carrying the address of the previous letter between recursions. That address is updated in two ways: if the letter already exists, *head* gets the value of the existing pointer, otherwise it gets the value of the pointer to the newly created node.

The search algorithm searches for a search key *v* (a string) that is created by the spatio-temporal module. The algorithm consists of a main function `Search` that initializes the variables *position*, *head*, and *upLink*, copies the string *v* to another string *m* that will carry the actual search path (the adaptive path), and calls the adaptive search function `SearchR`.

The search operation succeeds if it ends by finding a flag either directly or adaptively. The first part of the recursive search extracts a letter and finds its *order*. If the *order* is null, i.e. end of string, the function moves immediately to the next node (the leaf), retrieves the flag, and end the search procedure. On the other hand, if the letter is not null, the function checks for the existence of the extracted letter by reading the memory address in the location *head+order*. A null value indicates that the letter does not exist, otherwise the *upLink* variable gets the value of *head* and a new recursion is called. If the letter does not exist, the adaptive search is invoked.

Before performing adaptive search, the function checks if it is allowed in this node by reading the *adaptive-search* variable.

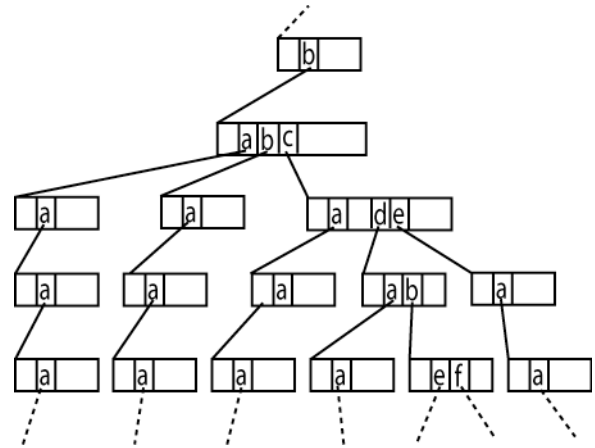


Figure 6. A portion of the ASM-trie

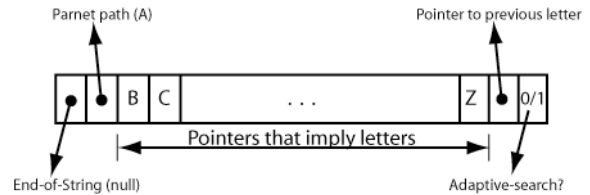


Figure 7. ASM-trie node

The adaptive search looks first for a parent path in the same node by looking for letter 'a'. If letter 'a' is found, the current letter is replaced by letter 'a' in the string *m* and the search continues from there. If there is no parent path in the same node, the procedure keeps retreating one letter at a time while searching for the parent path. During retreat, the *position*, *head* and *upLink* variables are adjusted to reflect the values in the current location. When a letter 'a' is found, the search proceeds with the rest of the original string and re-invokes adaptive search whenever is needed.

2.6 Comments on the System Design

As the next section shows, the adoption of a memory resident approach for the ASM-trie provides the clear advantage of faster performance over external memory structures. We also observe that the conceived limit on main memory size should not affect the scalability of the system for several reasons: 1) the LS is implemented as a distributed system where every node is responsible for a specific area, referred to as the service area [15, 16], 2) with 64-bit processors becoming a commonplace even for small servers, the physical limit on main memory is becoming less restrictive, and 3) the new trend in implementing large-scale services, e.g., Google search engine, is to rely on multiple cheap servers where all the data is indexed in the memory [3].

ALGORITHM 1 ASM-trie Insert.

INPUT: An Access Rule Class

OUTPUT: Success or Failure.

FUNCTION Insert $ars \leftarrow accessRule.getArs()$ $f \leftarrow accessRule.getF()$ Let $position, head, upLink \leftarrow 0$ Let $IsAdaptiveSearch \leftarrow \mathbf{FALSE}$ Call **InsertR** ($head, upLink, ars, f, position$)**FUNCTION** InsertRExtract a new letter from ars **IF** the letter is lowercase **THEN** $IsAdaptiveSearch \leftarrow \mathbf{TRUE}$ **ENDIF**Convert the letter to an $order$ value**IF** the $order$ value is **NULL** **THEN****IF** there is an existing flag **THEN**Return **FAILURE****ELSE**insert flag f Return **SUCCESS****ENDIF****ELSE****IF** the letter does not exist in that path **THEN**Create a new node and assign its memory address to pointer $head$ $previous-letter \leftarrow upLink$ $adaptive-search \leftarrow IsAdaptiveSearch$ **ELSE****IF** the letter exists **THEN**Assign value of current pointer to $head$ **ENDIF****ENDIF** $upLink \leftarrow head$ Call **InsertR** ($head+order, upLink, ars, f, position+1$)**ENDIF**

ALGORITHM 2 ASM-trie Search.

INPUT: A Search Key String

OUTPUT: A Flag.

FUNCTION SearchLet $position, head, upLink \leftarrow 0$ $m \leftarrow v$ Call **SearchR** ($head, upLink, ars, f, position$)**FUNCTION** SearchRExtract a new letter from the string v Convert the letter to an $order$ **IF** the $order$ value is **NULL** **THEN**

Retrieve the flag

Return the flag

ELSERead pointer at memory address = $head + radix$ **IF** pointer value is **NULL** **THEN****WHILE** **NOT** at beginning of v

{invoke adaptive search}

 $position \leftarrow position - 1$ $head \leftarrow upLink$ $upLink \leftarrow head.previous-letter$ **IF** **NOT** $adaptive-search$ **THEN**Return **FAILURE****ELSE****IF** there a parent path **THEN** $position \leftarrow position+1$ $upLink \leftarrow head$ $head \leftarrow upLink.next[1]$ Update strin m with new pathBreak the **WHILE** loop**ENDIF****ENDIF****ENDWHILE** $upLink \leftarrow head$ Call **SearchR** ($head+order, upLink, v, position+1$)**ENDIF**

3. EXPERIMENTAL STUDY

We conducted a performance evaluation study using randomly generated sets of access rules and search keys. The random values were drawn from actual hierarchies of industry, location, and time, and then encoded as strings. For each graph in Figure 7, each point represents an overall average of 30 replicas of access rules and 100 replicas of search key sets at 10 different sizes. The results are presented on a logarithmic scale.

We conducted the experiment for three different approaches: the ASM-trie, a regular trie structure with linear search, and an Oracle based search. The machine used for the experimentation is Intel-based with dual Pentium 4 Xeon processors running at 2.4 GHz, and 2GB of RAM. The ASM-trie outperformed the two other techniques. It exhibited a constant search time, around 32000 keys/Sec for that experiment, while the regular trie had a constant time around 173 keys/Sec. Oracle based approach had around 80 keys/ Sec for small databases, however, it deteriorated with large samples to 20 keys/Sec. The ASM-trie also has sub-linear memory utilization around 1200 rules/MB in this experiment. This number directly depends on the length of the encoded string of the access rule. Notice that large percentage of the nodes in the trie are shared among several access rules.

Since the only difference between the ASM-trie and the regular trie is the adaptive search, it is reasonable to conclude that the positive difference in performance is due to the adaptive search.

4. RELATED WORK

Schilit et al. [18] proposed to apply access control on the customer device. This approach has several disadvantages: first, the customer will receive all the offers, and hence will bear the cost for receiving them. Second, as the authors admit, this approach is not appropriate for large number of offers since it can create much traffic for the network. Third, it requires the network collaboration in deciding who should receive the offers which implies that network is trusted for the customer location information. Hengartner et al. [12, 13] proposed an approach for protecting people location in ubiquitous computing in the context of an organization such as a hospital or a university which is different from our settings in terms of geographical coverage, the number and homogeneity of objects and subjects, and location modeling. Hauser et al. [10, 11] discussed the problem of privacy protection in the context of the NEXUS project which has very similar setting to ours. Their solution focuses on protecting privacy using anonymity techniques.

Bertino et al. [5] proposed a spatial access control model for Web Map Management application. The model addresses the spatial constraints by adding a valid geographical scope to authorizations and defines new access modes such as *notify*. Nevertheless, the model does not support temporal constraints or moving objects. There are also several advanced access control models that support temporal, spatial, and spatio-temporal operations, however they are not appropriate for the addressed problem. This includes: the *Temporal Authorization Model* (TAM), the *Temporal Data Authorization Model* (TDAM), the *Digital Library Authorization Model* (DLAM), and the *Digital Library Authorization System* (DLAS) [1].

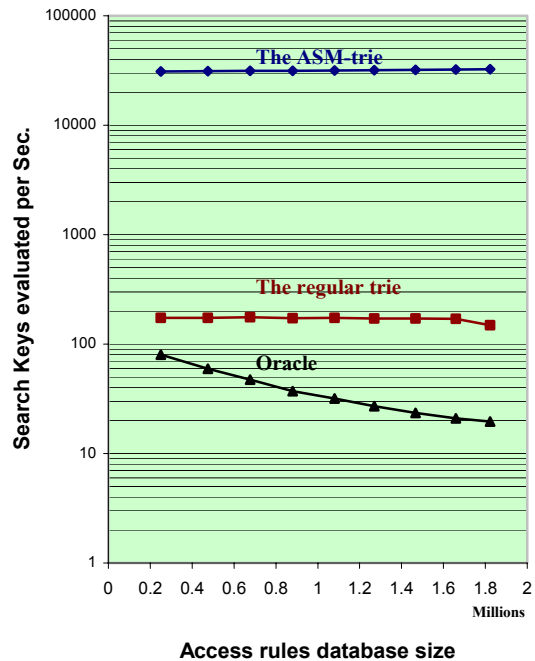


Figure 7. Search performance among ASM-trie, regular trie, and Oracle

5. SUMMARY AND FUTURE WORK

We presented a solution to improve customer privacy in mobile services. The solution includes a design of information flow that prevents merchants from sharing and aggregating customer information, an access control model for moving objects and customer profiles that supports granular representation, and an efficient enforcement mechanism. The model design contributes to the efficiency and scalability of the enforcement mechanism by enabling adaptive search and supporting positive and negative permissions. The mechanism utilizes a new data structure, the ASM-trie; and a spatio-temporal module that supports interaction between moving objects and spatio-temporal constraints and translates between geospatial coordinates and civil names. Due to space limitation, we have left out discussion on model evolution and analysis of the algorithms.

We are working on extending this research in several directions. One of these directions is to build a disk-based version of the ASM-trie and compare it to the memory-based version. The other direction is to semantically enhance the access control system by utilizing ontologies that capture the semantics in the hierarchies.

6. REFERENCES

- [1] N. R. Adam, V. Atluri, E. Bertino, and E. Ferrari, "A Content-Based Authorization Model for Digital Libraries", In *IEEE TKDE 14(2)*, 2002, pp. 296-315.
- [2] N. Asokan, P. A. Janson, M. Steiner, and M. Waidner, "State of the art in electronic payment systems", In *IEEE Computer*: 30(9) pp. 28-35, 1997.

- [3] E. Bertino, P. Samarati, S. Jajodia, "An extended authorization model for relational databases," In *IEEE TKDE* (9)1, 1997, pp. 85-101.
- [4] L. A. Barroso, J. Dean, and U. Hölzle, "Web Search for a Planet: The Google Cluster Architecture", In *IEEE micro*, March 2003, pp. 22-28.
- [5] E. Bertino, M. L. Damiani, and D. Momini, "An Access Control System for a Web Map Management Service", in *Proceedings of (RIDE'04)*, Boston, 2004, pp. 33-39.
- [6] R. K. Chellappa and R. Sin "Personalization versus privacy: An empirical examination of the online consumer's dilemma", In *Infirms Meeting*, 2002.
- [7] K. Elbassioni, A. Elmasri, and I. Kamel, An Efficient Indexing Scheme for Multi-dimensional Moving Objects, In *Proceedings of ICDDT2003*, Siena, Italy, January 2003.
- [8] E. Fredkin, "Trie Memory", In *Communications of the ACM*, September 1960, pp. 490-500.
- [9] G. Gow, "Pinpointing Consent: Location Privacy, Public Safety, and Mobile Phones", In *the Conference on the Global and Local in Mobile Communications*, Budapest, Hungary, June, 2004.
- [10] C. Hauser, "Privacy and Security in Location-Based Systems With Spatial Models", In *the PAMPAS workshop*, Royal Holloway, University of London, September 2002.
- [11] C. Hauser, and M. Kabatnik, "Towards privacy support in a global location service", In *Proceedings of the IFIP Workshop on IP and ATM Traffic Management*, Paris, 2001, pp. 81-89.
- [12] U. Hengartner, and P. Steenkiste, "Protecting Access to People Location Information", In *Proc. of First International Conference on Security in Pervasive Computing (SPC 2003)*, Boppard, Germany, March 2003, pp. 25-38.
- [13] U. Hengartner, and P. Steenkiste, "Implementing Access Control to People Location Information", In *Proc. of SACMAT'04*, Yorktown Heights, NY, June 2004, pp.11-20.
- [14] C. Jensen, "research challenges in location-enabled M-services", In *Proceedings of the third IEEE MDM*, Singapore, January 2002.
- [15] G. Kollios, D. Gunopulos, and V. J. Tsotras, "On indexing mobile objects", In *Proceedings of PODS*, 1999.
- [16] U. Leonhardt and J. Magee, "Security Considerations for a Distributed Location Service", In *Journal of Network and Systems Management*, 6(1), March 1998, pp. 51-70.
- [17] A. Leonhardt, and K. Rothermel, "Architecture of a Large-scale Location Service", In *Proceedings of ICDCS'02*, Vienna, Austria, July 2002.
- [18] B. Schilit, J. Hong, and H. Gruteser, "Wireless Location Privacy Protection", In *IEEE Computer Magazine*, December 2003.
- [19] S. Saltis, C. S. Jensen, S. Leutenegger and M. Lopez, "Indexing the Positions of Continuously Moving Objects", In *Proceedings of the 19th ACM-SIGMOD*, Dallas, Texas, 2000.
- [20] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects", In *Proceedings ICDE'97*, Birmingham, UK, 1997.
- [21] J. Tayeb, O. Ulusoy, and O. Wolfson, "A quadtree based dynamic attribute indexing method", In *Computer Journal*, 41(3), 1998, pp. 185-200.
- [22] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving objects databases: Issues and solutions", In *Proceedings of the 10th ICSSDM*. Capri, Italy, July 1998.
- [23] UTMSWORLD.COM, "UMTS Location Based Services", <http://www.umtsworld.com/technology/lcs.htm>, June 2004.