

Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

8-2004

LTAM: A Location-Temporal Authorization Model


Hai YU

Ee Peng LIM

Singapore Management University, eplim@smu.edu.sg

DOI: https://doi.org/10.1007/978-3-540-30073-1_13

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Databases and Information Systems Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

Citation

YU, Hai and LIM, Ee Peng. LTAM: A Location-Temporal Authorization Model. (2004). *Secure Data Management: VLDB 2004 Workshop, SDM 2004, Toronto, Canada, August 30, 2004. Proceedings*. 3178, 172-186. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/1022

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

LTAM: A Location-Temporal Authorization Model

Hai Yu and Ee-Peng Lim

Center for Advanced Information Systems,
Nanyang Technological University, Singapore
yuhai@pmail.ntu.edu.sg, aseplim@ntu.edu.sg

Abstract. This paper describes an authorization model for specifying access privileges of users who make requests to access a set of locations in a building or more generally a physical or virtual infrastructure. In the model, primitive locations can be grouped into composite locations and the connectivities among locations are represented in a multilevel location graph. Authorizations are defined with temporal constraints on the time to enter and leave a location and constraints on the number of times users can access a location. Access control enforcement is conducted by monitoring user movement and checking access requests against an authorization database. The authorization model also includes rules that define the relationships among authorizations. We also describe the problem of finding inaccessible locations given a set of user specified authorizations and a multilevel location graph, and outline a solution algorithm.

1 Introduction

Access control is an important aspect of computer security. It provides a framework for protecting resources within a system by restricting the accesses to *objects* (or resources) by *subjects* (or users). Other than objects and subjects, a basic access control model consists of *rules* that govern the way subjects are granted accesses to objects. Access control models can be discretionary or mandatory. In *discretionary access control* (DAC), owners of objects may grant access to others and are responsible for protecting the objects they own. In contrast, *mandatory access control* (MAC) assigns each object a security label that is used as the basis of restricting accesses of the users to the object. DAC has been widely adopted by commercial applications and databases systems. Due to its rather constrained way of granting access, the use of MAC has not been popular among commercial applications.

As wireless devices (e.g., RFIDs, handphones) become ubiquitous and are often equipped with positioning capabilities, they have been increasingly used for tracking user and object movements to support a wide range of applications[1–3]. For example, Singapore has used RFIDs to track movements of hospital users during the outbreaks of SARS (Severe Acute Respiratory Syndrome), a highly contagious and deadly disease. From the user movement data, users who were in

contact with diagnosed SARS patients could be traced and placed in quarantine or observations[4].

In homeland security, preventive measures are highly critical. As part of the efforts to safeguard the security of physical infrastructure, movements of users within a secured building can be tracked and their accesses to various locations in the building can be controlled by a security system that supports flexible access control. The ability of user tracking is also assumed in this research on authorization model.

In this paper, we propose a location-temporal authorization (LTAM) model that allows locations to be treated as objects and user accesses to these locations are restricted. The enforcement of such an authorization model requires maintaining the current locations of users and processing their access requests. Based on this model, computation and reasoning can be conducted on the authorizations to derive useful properties and knowledge about the location and time where authorizations are given.

Our proposed LTAM model differs from the existing office security systems that involve the use of card readers to authenticate and register user access requests for entering a room. The key differences are:

- The existing systems only enforce access control upon access requests while LTAM monitors the user movement at all times. This eliminates situation where a group of users enters a restricted location based on a single user authorization.
- LTAM can support more expressive access control restrictions. For example, one may be authorized to leave a location only during a certain time interval. Should this restriction be violated, security alerts can be triggered.
- LTAM can support an interesting range of queries on the authorizations and these queries are necessary to implement applications that manage movement and accesses to locations in a secure infrastructure. This is clearly a large improvement over the existing ad-hoc implementations.
- LTAM provides a framework for analyzing the security shortfalls due to human errors in specifying authorizations.
- LTAM protects the location privacy [5] of the users by restricting the location information in the central control station and not releasing it to other applications.

1.1 Outline of Paper

The rest of the paper is organized as follows. Section 2 reviews the existing work in temporal authorization models and context-aware information security. Our proposed authorization model is defined in Section 3 followed by the enforcement of the model described in Section 5. The authorization rules that allow new authorizations to be derived will be defined in Section 4. The problem of finding inaccessible locations and its solution are given in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Our proposed location-temporal authorization model falls under the area of spatio-temporal access control. Our literature survey however has found very little research work on this topic. We therefore examine some of the related work in temporal access control and spatial access control.

One of the first papers about temporal authorization model came from Bertino, Bettini and Samarati[6]. In their proposed authorization model known as TAM, each authorization for a user to access an object is augmented with a temporal interval of validity. In other words, the user is only able to access the object during the specified temporal interval and the dependencies among temporal authorizations can be specified within the proposed model. In [7], Gal and Atluri proposed another temporal authorization model called TDAM to support discretionary access control based on the temporal attributes of the objects themselves. Both TAM and TDAM are complementary models and can be used together.

An authorization model that specifically addresses access control issues of geospatial objects was proposed by Atluri and Mazzoleni[8]. This model known as GSAM can authorize users to view specific *region* within a satellite image object with a certain *resolution*. An indexing structure supporting efficient retrieval and enforcement of GSAM authorizations on satellite image has been developed. GSAM however does not include spatial locations of users and temporal dimension in the specification of authorizations.

In the area of pervasive computing, context aware role-based access control was proposed to model transitions of user roles and object states due to contextual changes and to grant users access privileges to objects based on the context at the time of access requests[9]. This proposed model however does not include the temporal and location dimensions of authorizations. Jiang and Landay further defined the notion of *information space* to organize information objects and services into different boundaries for better privacy control[10]. The boundaries can be defined by physical space, social grouping, or activity. By granting access privileges differently for different information spaces, authorizations can be made more context aware. We believe that information space can be viewed as some kind of locations in our proposed authorization model. Using our proposed model, information spaces can be linked together representing their relationships, and users are required to be authorized before entering an information space or moving from one information space to another.

Finally, a location and user authentication architecture was given in [11]. The paper however did not provide a comprehensive model to represent authorizations that involve both time and locations.

3 Location-Temporal Authorization Model

In this section we describe our Location-Temporal Authorization model (LTAM) in detail.

3.1 Preliminaries

Locations in LTAM are both *semantic* and *physical*. When represented physically, a location is described by its absolute spatial coordinates. In [12], Pradhan describes semantic locations as objects with unique identifiers so as to give semantic meanings to the locations. The physical location information are used to define the spatial boundaries of location so that it is possible to track users in different locations. A location can be *primitive* or *composite*. A *primitive location* is a location that cannot be further divided into other smaller locations. A *composite location* is a collection of related primitive, composite, or a mix of both locations. For instance, a room in a building is a primitive location, and the building which consists of a number of rooms is a composite location. All rooms in the building forms a *location graph* that represents the building. The building together with other buildings form a *multilevel location graph*. Formally, we define location graph and multilevel location graph as follows.

Definition 1 (Location Graph). *A location graph is defined as (L,E) where*

- L is a set of primitive locations
- E is a set of edges connecting pairs of locations

Within a location graph, if (l_1,l_2) is an edge e , it implies that l_2 can be reached from l_1 directly without going through other locations, and vice versa. By definition, an edge is bidirectional.

Definition 2 (Multilevel Location Graph).

If G_1, \dots, G_k are location graphs or multilevel location graphs with mutually disjoint locations, then (L', E) is a multilevel location graph where $L' = \{G_1, \dots, G_k\}$ and $E \subseteq L' \times L'$

Each location graph or multilevel location graph must have at least one location designated as *entry location*. An entry location serves as the first location a user must visit before visiting other locations within the graph. A entry location also serves as the last location where the user may visit before his/her exit. In some cases it is possible that the entry and exit locations have to be treated separately, which we have not considered in this paper. We believe our proposed model can be easily extended to deal with these cases.

Let H be a multilevel location graph and l_i be a primitive location (or composite location), we say that l_i is *part of H* if l_i is a primitive location (or composite location) that directly or indirectly belongs to H .

Fig. 1 depicts the location layout of School of Computer Engineering and School of Electrical and Electronic Engineering in Nanyang Technological University. Fig. 2 shows the corresponding multilevel location graph, where NTU is a multilevel location graph and SCE, EEE, CCE, SME, NBS¹ are all location graphs. The locations with double lines denote the entry locations.

¹ SCE, EEE, CCE, SME, NBS are the schools in the university

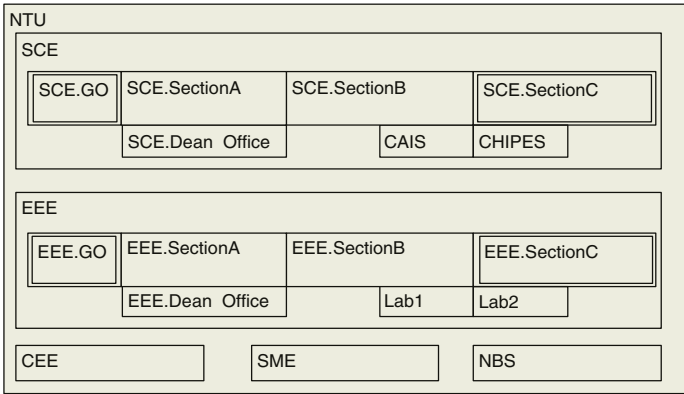


Fig. 1. A Location Layout

In Fig. 2, primitive locations SCE.GO, SCE.Dean’s Office, CAIS, CHIPES, SCE.SectionA, SCE.SectionB, SCE.SectionC² form a location graph named SCE. The entry locations of SCE are SCE.GO and SCE.SectionC. To access any location that is part of SCE, one has to go through at least one of these two entry locations. The edge between SCE.SectionB and SCE.CAIS shows one to go from SCE.SectionB to CAIS directly and vice versa.

A *simple route* in a location graph, (G, E) , refers to a series of primitive locations $\langle l_1, l_2, \dots, l_k \rangle$ (l_i 's $\in G$) through which a subject can move from location l_1 to location l_k , i.e., $(l_i, l_{i+1}) \in E, \forall 1 \leq i < k$. For example, $\langle \text{SCE.Dean's Office, SCE.SectionA, SCE.SectionB, CAIS} \rangle$ is a simple route.

A *complex route* in a multilevel location graph (G, E) refers to a series of primitive locations $\langle l_1, l_2, \dots, l_k \rangle$ through which a subject can move from l_1 to location l_k such that $\forall 1 \leq i < k$,

- (l_i, l_{i+1}) is an edge in some location graph; or
- l_i and l_{i+1} are entry locations in two different location graphs G_i and G_{i+1} respectively. G_i and G_{i+1} are multilevel location graphs of two composite locations l'_i and l'_{i+1} , respectively, such that (l'_i, l'_{i+1}) is an edge in some multilevel location graph G' that contains both G_i and G_{i+1} .

For example in Fig. 2, $\langle \text{EEE.Dean's Office, EEE.SectionA, EEE.GO, SCE.GO, SCE.SectionA, SCE.Dean's Office} \rangle$ is a complex route.

In a route $r, \langle l_1, l_2, \dots, l_n \rangle$, l_1 and l_n are called the *source* and the *destination* of r , respectively. Note that there can be multiple routes from a source to a destination.

Location graphs are connected graphs. For a given location graph (L, E) , there exist a route r such that l_d can be reached from l_s , for any $l_s, l_d \in L$. Similarly multilevel location graphs are also connected graphs.

² SCE.GO denotes the general office of SCE. CAIS and CHIPES are research centers in SCE

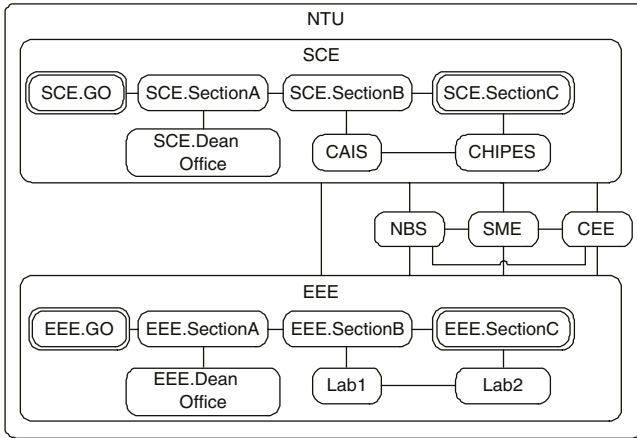


Fig. 2. A Multilevel Location Graph

Time is another important concept in the access control. We adopt the approach similar to that in [6]. A time unit is a *chronon* or a fixed number of chronons, where a chronon refers to the smallest invisible unit of time. A *time interval* is a set of consecutive time units. The size of the time interval is the number of time units in the time interval.

3.2 Location-Temporal Authorization

Location Authorizations are policies created by security officers for defining the accesses that the users have over the locations. *Location-Temporal authorizations* are location authorizations augmented with temporal conditions to limit the period during which the authorization is valid. Formally, they are defined as follows.

Definition 3 (Location Authorization). A location authorization is a pair (s, l) where

- s is a subject (user) who requests authorizations; and
- l is a primitive location

A location authorization (s, l) means that user s is authorized to enter the primitive location l . For example, $(\text{Alice}, \text{CAIS})$ denotes that Alice is authorized to access location CAIS.

Definition 4 (Location-Temporal Authorization). A location-temporal authorization is a quadruple (entry duration, exit duration, auth, entry) where

- entry duration is a time interval $[t_s^i, t_e^i]$ during which a subject can enter a primitive location
- exit duration is a time interval $[t_s^o, t_e^o]$ during which a subject can leave a primitive location, where $t_s^o \geq t_s^i$ and $t_e^o \geq t_e^i$

- *auth* is an location authorization
- *entry* is the number of accesses that the subject can exercise within entry duration. The range of entry is $[1, \infty)$.

A Location-Temporal Authorization imposes temporal constraints on a location authorization. An authorization $([t_1^i, t_2^i], [t_1^o, t_2^o], (\mathbf{s}, \mathbf{l}), \mathbf{n})$ indicates that user \mathbf{s} is authorized to enter primitive location \mathbf{l} during $[t_1^i, t_2^i]$ and exit during $[t_1^o, t_2^o]$, for a maximum number of \mathbf{n} times. If the entry duration is not specified, it means the subject can enter a location at any time after the creation of the authorization. On the other hand, if the exit duration is not specified, the default value will be $[t_1^o, \infty]$ which means that the subject can exit any time after entering the location. The default entry value is ∞ .

Consider the authorization $([5, 40], [20, 100], (\text{Alice}, \text{CAIS}), 1)$. Alice is allowed to enter location CAIS once during the period $[5, 40]$, and to exit during the period $[20, 100]$. If she does not exit CAIS during the exit duration, a warning signal to the security guards will be generated.

4 Authorization Rules

In large organizations, it is impractical to define authorizations for individual users on every location. In addition, some authorizations may only be valid when certain conditions are satisfied. Manually specifying all the authorizations is a very tedious and error-prone job. *Authorization rules* are therefore introduced to automate the work of deriving additional authorizations based on the existing authorizations. An authorization rule can also be viewed as a kind of relationship between authorizations. An authorization rule generates a number of authorizations based on an input authorization. The input authorization is called the *base authorization*. The generated authorizations are called the *derived authorizations*. The formal definition of authorization rule is as follows.

Definition 5 (Authorization Rule). *An authorization rule is defined as $\langle t_r : (a, OP) \rangle$, where*

- t_r is the time from when the authorization rule is valid.
- $a = ([t_s^i, t_e^i], [t_s^o, t_e^o], (s, l), n)$ is the base authorization
- OP is a tuple of operators $(op_{\text{entry}}, op_{\text{exit}}, op_{\text{subject}}, op_{\text{location}}, exp_n)$, where
 - op_{entry} and op_{exit} are temporal operators, which take $[t_s^i, t_e^i]$ and $[t_s^o, t_e^o]$ of a as inputs, and generate the entry and exit durations for the derived authorizations, respectively.

The temporal operators can be one of the following:

- * WHENEVER

WHENEVER is a unary operator which returns the same time interval as the input.

- * WHENEVERNOT

Given an input time interval, $[t_0, t_1]$, the unary operator WHENEVERNOT operator returns $[t_r, t_0 - 1]$ and $[t_1 + 1, \infty]$.

* UNION

UNION is a binary operator. Given two input time intervals $[t_0, t_1]$ and $[t_2, t_3]$, UNION returns $[t_0, t_3]$ if $t_2 \leq t_1$; or $[t_0, t_1]$ and $[t_2, t_3]$ if $t_2 > t_1$.

* INTERSECTION

INTERSECTION is a binary operator. Given two input time intervals $[t_0, t_1]$ and $[t_2, t_3]$, INTERSECTION returns $[t_2, t_1]$ if $t_2 \leq t_1$; Otherwise it returns NULL.

- op_{subject} takes subject s of a , and derives the subjects for the derived authorizations based on some relationships between subjects.
- op_{location} is a location operator, which generates a set of primitive locations for the derived authorizations, given the primitive location l of a .
- exp_n specifies a numeric expression on the number of entries.

If any of the rule elements is not specified in a rule, the default value will be copied from the base authorization.

Example 1. Consider the following authorization.

a1: ($[5, 20]$, $[15, 50]$, (Alice, CAIS), 2)

If we want the supervisor of Alice to have the same authorization on CAIS as that of Alice, we can define the following rule.

r1: $\langle 7:\mathbf{a1}, (\text{WHENEVER}, \text{WHENEVER}, \text{Supervisor_Of}, \text{CAIS}, 2) \rangle$

The op_{subject} operator **Supervisor_Of** returns the supervisor of a user by querying the *user profile database* described in the next section. Suppose Alice's supervisor is Bob, the following authorization can be derived.

a2: ($[5, 20]$, $[15, 50]$, (Bob, CAIS), 2)

By specifying this rule, it is not necessary to create new authorizations if Alice is assigned a different supervisor. The system is able to automatically derive the authorizations for the new supervisor while the authorization for Bob will be revoked.

Example 2. If we modify rule **r1** slightly as follows.

r2: $\langle 7:\mathbf{a1}, (\text{INTERSECTION}([10, 30]), \text{WHENEVER}, \text{Supervisor_Of}, \text{CAIS}, 2) \rangle$

The derived authorization of **r2** is

a3: ($[10, 20]$, $[15, 50]$, (Bob, CAIS), 2)

Rule **r2** specifies that the supervisor of Alice is supposed to access CAIS during $[10, 30]$, however, only when Alice is also authorized to access CAIS.

Example 3. Now given authorization `a1`, we would like to grant `Alice` access to all locations on the route from `SCE.GO` to `CAIS`. The following authorization rule can be specified for this purpose.

```
r3:(7:a1,(WHENEVER,WHENEVER,,all_route_from(SCE.GO),2))
```

The location operator `all_route_from` returns all the locations on the route from source `SCE.GO` to destination `CAIS`, which are `{SCE.GO, SCE.SectionA, SCE.SectionB, SCE.SectionC, SCE.CHIPES}`. An authorization will be derived for each of these locations as the result of rule `r2`.

Besides the operators aforementioned, customized operators can be defined as well, which leads to greater degree of flexibility.

It is worth noting that the authorization rules may introduce conflicts of authorizations, which means the derived authorizations may contradict with other authorizations. For example, a derived authorization may say that `Alice` can enter `CAIS` during `[5,10]`. However, another authorization (either existing or derived) may state that `Alice` is authorized to enter `CAIS` during `[10,11]`. This conflict should be resolved either by combining the two authorizations, or discarding one of them. The problem is left for future work.

5 Location-Temporal Authorization Enforcement

The authorizations are checked when an *access request* is posed by a subject. Formally, we define access request as follows.

Definition 6 (Access Request). *An access request is a triple (t, s, l) where*

- t is the time instant at which the access request is made
- s is the user who requests the access
- l is the location where the user requests to access

For example, a triple `(10,Alice,CAIS)` denotes that at time 10, `Alice` issued an access request to location `CAIS`.

An access request is checked against the set of authorizations in the system. If an authorization exists at time t , the access request is authorized. We define *authorized access request* as follows.

Definition 7 (Authorized Access Request). *An access request (t, s, l) is authorized if there exists at least one location temporal authorization*

$A : ([t_s^i, t_e^i], s, l, [t_s^o, t_e^o], n)$ *such that*

- $t_s^i \leq t \leq t_e^i$
- s has entered l during $[t_s^i, t_e^i]$ for less than n times.

For example, suppose that the system contains the following authorizations.

- `A1: ([10,20],[10,50],[Alice,CAIS],2)`
- `A2: ([5,35],[20,100],[Bob,CHIPES],1)`

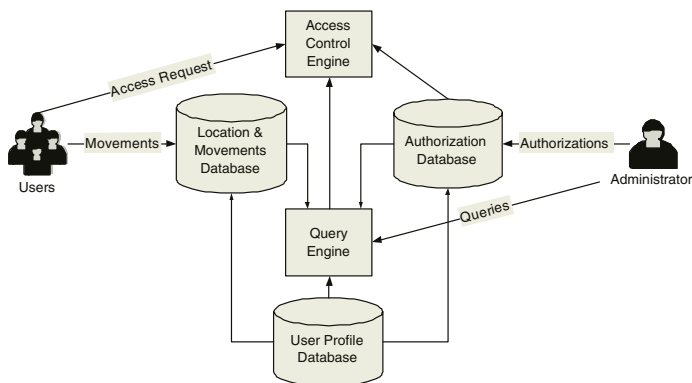


Fig. 3. System Architecture for Authorization Enforcement

Assume that each subject has not entered any location yet, we have

- At time 10, access request (10,Alice,CAIS) is granted according to A1.
- At time 15, access request (15,Bob,CAIS) is not authorized because there is no authorization specifies Bob’s access to CAIS.
- At time 16, access request (15,Bob,CHIPES) is authorized based on A2.
- At time 20, Bob leaves CHIPES.
- At time 30, access request (30,Bob,CHIPES) is not authorized because Bob has only one entry to CHIPES.

Fig. 3 shows the system architecture for location-temporal authorization enforcement. The system has five major components.

- **Authorization Database**
The authorization database stores all authorizations defined by the system administrators.
- **Location & Movements Database**
The location & movements database stores the location layout, as well as users’ movements. These data are then used for authorization validation, system status checking, etc..
- **User Profile Database**
As its name indicates, the user profile database stores user profiles, which are used for creating authorizations, or deriving authorizations, etc..
- **Access Control Engine**
The access control engine is the core of the authorization enforcement. When a user issues an access request, the access control engine have to perform a few tasks.
 1. It checks the authorization database to search for any authorization that has been defined for the user and the location that the user request access to.
 2. It invokes the query engine to find out whether the user has violated any authorization due to unauthorized access requests or over-staying.

3. Access control engine is also responsible for authorization derivation. When the administrator specifies new rules, the access control engine will evaluate the new rules on the existing authorizations and user profiles. The derived authorizations are then added to the authorization database.
- **Query Engine**
The query engine evaluates queries by the system administrators and the access control engine based on the information stored in all of the databases.

The design of a query language for our proposed authorization model will be part of our future work. Some of these questions can be complex. In the following section, we will present a query that find all locations inaccessible (or accessible) to a given subject.

6 Finding Inaccessible Locations

Given a set of LTAM authorizations, one can query and conduct reasoning or computation on them to derive useful knowledge. In this section, we will describe the problem of finding inaccessible locations and develop the corresponding solution algorithm.

Given an access request duration $[t_p, t_q]$ from a user s to a location l and a location-temporal authorization $a = ([t_s^i, t_e^i], [t_s^o, t_e^o], (s, l), entry)$, the *grant duration* of s for l in the access request duration is defined by $[\max(t_p, t_s^i), \min(t_q, t_e^i)]$, and the *departure duration* of s for l in the access request duration is defined by $[\max(t_p, t_s^o), t_e^o]$.

A route $r = \langle l_1, l_2, \dots, l_k \rangle$ is *authorized* for a subject s with access request duration $[t_p, t_q]$ if,

- The grant duration of s for l_1 in $[t_p, t_q]$, denoted by $[t_{p_1}^g, t_{q_1}^g]$, is not null;
- The departure duration of s for l_1 in $[t_p, t_q]$, denoted by $[t_{p_1}^d, t_{q_1}^d]$, is not null;
- The grant duration of s for l_i in $[t_{p_{i-1}}^d, t_{q_{i-1}}^d]$, denoted by $[t_{p_i}^g, t_{q_i}^g]$, is not null $\forall 2 \leq i < k$;
- The departure duration of s for l_i in $[t_{p_{i-1}}^d, t_{q_{i-1}}^d]$, denoted by $[t_{p_i}^d, t_{q_i}^d]$, is not null $\forall 2 \leq i < k$; and
- The grant duration of s for l_k in $[t_{p_{k-1}}^d, t_{q_{k-1}}^d]$, denoted by $[t_{p_k}^g, t_{q_k}^g]$, is not null.

The grant duration and departure duration of s for the route r are therefore $[t_{p_1}^g, t_{q_1}^g]$ and $[t_{p_k}^d, t_{q_k}^d]$ respectively.

Definition 8. *Given a subject s , a set of authorizations D and a location graph (or multilevel location graph) $G = (L, E)$, a location (or composite) l is known to be inaccessible by s if there is no authorized route for s with an access request duration $[0, \infty)$ that covers l from every entry location of G .*

Following the above definition, an entry location is inaccessible to a subject s if it has null exit duration for its authorization.

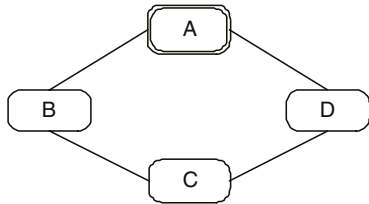


Fig. 4. An example of finding the inaccessible locations

From the above definition, we also know that a location can be made inaccessible to a subject by directly defining appropriate authorizations for that location, or by blocking all routes to the location. Hence, to ensure that a subject can visit a location, one should check that the location is not inaccessible instead of just defining the authorizations for that location.

The inaccessible location finding problem is thus defined as follows:

Definition 9. (*Inaccessible Location Finding Problem*) Given a subject s , a set of authorizations D and a location graph (or multilevel location graph) $G = (L, E)$, find all inaccessible locations in G .

We now outline a solution algorithm to the above problem. Our algorithm has been developed based on the following lemma which can be easily proven.

Lemma 1. Given a composite location l with a location graph or multilevel location graph (L, E) , if a location l' in L is inaccessible to a subject s considering only the entry locations in L , then the location l' is also inaccessible to s from every entry location in the multilevel location graph containing l .

The inaccessible location finding algorithm is shown in Algorithm 1. The algorithm first associates to each location l an overall grant time and a overall departure time, denoted by T^g and T^d respectively. Each of them consists of a set of time intervals. The overall grant time of each location is initialized to be null. As the algorithm assigns a location a new overall grant time, a new overall departure time is derived and the neighboring locations will adjust their overall grant and departure times accordingly. To indicate whether a location should be assigned a new overall grant and departure time, a boolean flag (denoted by *flag*) is associated with every location.

For example, consider the location graph in Fig. 4, consisting of locations A, B, C, and D, where A is the entry location. Suppose that a number of location-temporal authorizations have been defined for these locations as shown in Table 1. The steps of finding the inaccessible locations are shown in Table 2

The algorithm starts from the entry location A, by setting its grant duration T_A^g to $[2, 35]$ and departure duration T_A^d to $[20, 50]$. In the next step, its neighboring locations B and D are to be examined since their flags are set to true. B's grant duration T_B^g is assigned $[\max(20, 40), \min(50, 60)] = [40, 50]$ and its departure duration T_B^d is assigned $[\max(20, 55), 80] = [55, 80]$. Similarly, we can obtain D's grant duration T_D^g and the departure duration T_D^d , which are $[20, 25]$

Algorithm 1 FindInaccessible(G, s)**Input:** location graph $G = (L, E)$, subject s **output:** set of inaccessible locations

```

1: initialise  $l.T^g := l.T^d := null$  and  $l.flag := false$  for each  $l \in L$ 
2: for each entry location  $l_{\text{entry}} \in L$  do
3:   for each location-temporal authorization  $a$  of  $l_{\text{entry}}$  do
4:      $l_{\text{entry}}.T^g := l_{\text{entry}}.T^g \cup [a.t_s^i, a.t_e^i]$ 
5:      $l_{\text{entry}}.T^d := l_{\text{entry}}.T^d \cup [a.t_s^o, a.t_e^o]$ 
6:   end for
7:    $l_{\text{entry}}.flag := false$  // their admissible time will not change further
8:   if  $l_{\text{entry}}.T^d \neq null$  then
9:     for each  $l$  next to  $l_{\text{entry}}$  do
10:       $l.flag := true$ 
11:    end for
12:   end if
13: end for
14: while  $\exists l \in L$  where  $l.flag = true$  do
15:   for each  $l \in L$  where  $l.flag = true$  do
16:      $l.flag := false$ 
17:      $l.T^{old,d} := l.T^d$ 
18:      $T := \cup_{l_i \text{ next to } l} l_i.T^d$ 
19:     for each  $[t_p, t_q] \in T$  do
20:       for each location-temporal authorization  $a$  of  $l$  do
21:          $t := [\max(t_p, a.t_s^i), \min(t_q, a.t_e^i)]$ 
22:         if  $t \neq null$  then
23:            $l.T^g := l.T^g \cup t$ 
24:            $l.T^d := l.T^d \cup [\max(t_p, a.t_s^o), a.t_e^o]$ 
25:         end if
26:       end for
27:     end for
28:     if  $l.T^d \neq l.T^{old,d}$  then
29:       for each  $l'$  next to  $l$  do
30:          $l'.flag := true$ 
31:       end for
32:     end if
33:   end for
34: end while
35: Return  $\{l | l \in L \text{ and } l.T^g = null\}$ 

```

and $[20, 30]$, respectively. After processing B and D, the flags of A and C are set to true because they are the neighbors of B and D. For C, both the grant duration T_C^g and the departure duration T_C^d are null. For A, it updates its T_A^g and T_A^d to $[2, 35] \cup [20, 35] = [2, 35]$ and $[20, 50] \cup [30, 50] = [20, 50]$, respectively, according to the new values of the grant and departure durations of its neighbors. Since there is no change to both durations, A will not update its neighbors. Therefore the whole process stops because no location has a flag set to true.

Table 1. A set of authorizations

Location	Authorization
A	$([2, 35], [20, 50], (\text{Alice}, \text{A}), 1)$
B	$([40, 60], [55, 80], (\text{Alice}, \text{B}), 1)$
C	$([38, 45], [70, 90], (\text{Alice}, \text{C}), 1)$
D	$([5, 25], [10, 30], (\text{Alice}, \text{D}), 1)$

Table 2. An illustration of the example

Location	A			B			C			D		
	flag	T_A^g	T_A^d	flag	T_B^g	T_B^d	flag	T_C^g	T_C^d	flag	T_D^g	T_D^d
Initiation	F	ϕ	ϕ	F	ϕ	ϕ	F	ϕ	ϕ	F	ϕ	ϕ
Update A	F	[2, 35]	[20, 50]	T	ϕ	ϕ	F	ϕ	ϕ	T	ϕ	ϕ
Update B	T	[2, 35]	[20, 50]	F	[40, 50]	[55, 80]	T	ϕ	ϕ	T	ϕ	ϕ
Update D	T	[2, 35]	[20, 50]	F	[40, 50]	[55, 80]	T	ϕ	ϕ	F	[20, 25]	[20, 30]
Update C	T	[2, 35]	[20, 50]	F	[40, 50]	[55, 80]	F	ϕ	ϕ	F	[20, 25]	[20, 30]
Update A	F	$[2, 35] \cup [20, 35] = [2, 35]$	$[20, 50] \cup [20, 50] = [20, 50]$	F	[40, 50]	[55, 80]	F	ϕ	ϕ	F	[20, 25]	[20, 30]

T – True F – False ϕ – null

The above algorithm has the time complexity of $O(N_L^2 \cdot N_d \cdot N_a)$ where N_L denotes the number of locations in L , N_d denotes the maximum degree of locations, and N_a denotes the maximum number of authorizations for each location. Though the complexity is of a relatively high order, it should not cause any problem considering the fact that the number of locations in a building is limited in most cases. Note that the algorithm covers the possibility that there may exist multiple routes between two locations, by considering the grant and departure durations of all neighbors of every location.

7 Conclusions

We have defined a new authorization model for granting accesses to locations with temporal considerations. This model, LTAM, can represent the location layout using a location graph or multilevel graph. By monitoring a user’s movement and evaluating location access requests against user specified authorizations, one can determine if the user can be granted access to a location and if the user should leave the location. We also describe based on the proposed model the interesting problem of finding inaccessible locations within a (multi-level) location graph given a database of authorizations. A solution algorithm that explores authorized routes to locations in a (multilevel) location graph has been developed.

As part of the future work, we plan to expand the location-temporal authorization definition to include more access constraints. More authorization rules will be explored to represent more expressive rules. The consistency issues among the rules will be studied. A query language and the corresponding query operators will also be studied. Lastly, we would like to further integrate other context

about data objects and subjects into our model to provide more comprehensive mechanisms to support applications with advanced security requirement.

References

1. Hightower, J., Borriello, G.: A survey and taxonomy of location systems for ubiquitous computing. *IEEE Computer* **34** (2001) 57–66
2. Pitoura, E., Samaras, G.: Locating objects in mobile computing. *Knowledge and Data Engineering* **13** (2001) 571–592
3. Awerbuch, B., Peleg, D.: Online tracking of mobile users. In: *Proceedings of the ACM SIGCOMM Symposium on Communication Architectures and Protocols*. (1991)
4. *RFiD Journal: Singapore fights SARS with RFID*. *RFiD Journal*, <http://www.rfidjournal.com/article/articleview/446/1/1/> (2003)
5. Beresford, A.R., Stajano, F.: Location privacy in pervasive computing. *IEEE Pervasive Computing* **2** (2003) 46 – 55
6. Bertino, E., Bettini, C., Samarati, P.: A temporal authorization model. In: *Proceedings of the 2nd ACM Conference on Computer and Communications Security (CCS '94)*. (1994) 126–135
7. Gal, A., Atluri, V.: An authorization model for temporal data. In: *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS2000)*. (2000) 144–153
8. Atluri, V., Mazzoleni, P.: A uniform indexing scheme for geospatial data and authorizations. In: *IFIP WG 11.3 Sixteenth International Conference on Data and Applications Security (DBSec2002)*. (2002)
9. Zhang, G., Parashar, M.: Context-aware dynamic access control for pervasive applications. In: *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS2004)*. (2004)
10. Jiang, X., Landay, J.A.: Modeling privacy control in context-aware systems. *IEEE Pervasive Computing* **1** (2002) 59–63
11. Michalakakis, N.: PAC: Location aware access control for pervasive computing environments. In: *MIT Student Oxygen Workshop*. (2002)
12. Pradhan, S.: Semantic location. HP, <http://cooltown.hp.com/dev/wpapers/semantic/sematnic.asp> (2002)