Supporting Location-Based Conditions in Access Control Policies

Claudio A. Ardagna
DTI - Università di Milano
26013 Crema - Italy
ardagna@dti.unimi.it

Marco Cremonini
DTI - Università di Milano
26013 Crema - Italy
cremonini@dti.unimi.it

Ernesto Damiani
DTI - Università di Milano
26013 Crema - Italy
damiani@dti.unimi.it

Sabrina De Capitani di Vimercati
DTI - Università di Milano
26013 Crema - Italy
decapita@dti.unimi.it

Pierangela Samarati
DTI - Università di Milano
26013 Crema - Italy
samarati@dti.unimi.it

ABSTRACT

Location-based Access Control (LBAC) techniques allow taking users' physical location into account when determining their access privileges. In this paper, we present an approach to LBAC aimed at integrating location-based conditions along with a generic access control model, so that a requestor can be granted or denied access by checking her location as well as her credentials. Our LBAC model includes a novel way of taking into account the limitations of the technology used to ascertain the location of the requester. Namely, we describe how location verification can be encapsulated as a service, representing location technologies underlying it in terms of two semantically uniform *service level agreement* (SLA) parameters called *confidence* and *timeout*. Based on these parameters, we present the formal definition of a number of location-based predicates, their management, evaluation, and enforcement. The challenges that such an extension to traditional access control policies inevitably carries are discussed also with reference to detailed examples of LBAC policies.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security

Keywords

Access control, location-based services, mobile system

1. INTRODUCTION

Conventional access control mechanisms rely on the assumption that requestors' profiles fully determine what they are authorized to do. However, more generally, a requestor's profile is not the only thing that matters: her physical location may also play an important role in determining access rights. *Location-based Access Control* (LBAC) is an intuitive concept: for example, being able to operate a mechanical device located in a particular room requires having a physical presence in that room. The very design of the device interface, including no remote control, is what enforces the security policy. Achieving the same kind of guarantee with software applications reachable via a telecommunication infrastructure like a wireless network requires a way to perform *location verification*, where a user's location is securely verified to meet certain criteria, for example, being inside a specific room or within a geographical area. The rapid development in the field of wireless and mobile networking fostered a new generation of devices suitable for being used as sensors by location technologies able to compute the relative position and movement of users in their working environment. Once a user's location has been verified using a protocol for location verification, the user can be granted access to a particular resource according to the desired policy. The location verification process must be able to tolerate rapid context changes, because mobile users can wander freely while initiating transactions by means of terminal devices like cell phones (GSM and 3G) and palmtops with wi-fi cards. To this end, well known location sensing techniques like the *Global Positioning System* (GPS) or techniques based on measuring signal power losses or transmission delays between terminals and wireless base stations can be exploited. Regardless of the specific technology, location verification can provide a rich context representation regarding both the users and the resources they access. Location-based information now potentially available to access control modules include the position of the requestor when a certain access request is submitted and the direction where she is headed. In the near future, location-based services will provide a wealth of additional environment-related knowledge (e.g., is the user sitting at her desk or walking toward the door? is she alone or together with others?). This kind of fine-grained context information potentially supports a new class of location-aware policy conditions regulating access to and fruition of resources.

When evaluating location-aware conditions, however, we need to consider that location-based information is radically different from other context-related knowledge inasmuch it is both *approximate* (all location systems have a margin of error) and *time-variant* (location is subject to fast changes, especially when the user is in motion). In this paper, we put forward the idea of integrating location-based conditions along with a generic access control model, so that a requestor could be granted or denied access by validating location-based credentials. We present the formal definition of some location-based predicates, their management, evaluation, and enforcement. The challenges that such an extension to traditional access control policies inevitably carries are discussed also with reference to detailed examples of LBAC policies. Our approach to LBAC includes a novel way of taking into account the specific techniques and algorithms used to ascertain the location of the requester. We represent the underlying location technologies in terms of a set of standard interfaces and semantically uniform *service level agreement* (SLA) parameters called *confidence* and *timeout*. Specifically, we describe the interface between an access policy evaluation engine and a Location Service. We then show how the SLA parameters of a Location Service can be taken into account by the Access Control Engine. Our solution fully addresses both uncertainty and time-dependency of location-based information; furthermore, it has the ability to seamlessly integrate location-based and identity-based access control, providing the exact level of security needed for pervasive and distributed resources. Finally, our architecture is highly distributed and relies on external Web services to perform functions like estimation of location of a resource requestor.

The remainder of this paper is organized as follows. Section 2 presents our location architecture and the basic concepts introducing a location-based scenario. Section 3 formally illustrates and defines location-based predicates. Section 4 describes access control policies based on location predicates, and Section 5 presents the working of the Access Control Engine enforcing LBAC policies. Section 6 presents related work. Finally, Section 7 gives our conclusions.

2. BASIC CONCEPTS AND REFERENCE SCENARIO

We briefly describe the reference location-based architecture and some basic concepts on location-based systems.

2.1 Location-based architecture

In a LBAC architecture, there are more parties involved than in conventional access control systems. Evaluation of LBAC policies involves context data about location and timing that are made available by third parties through service interfaces called *Location Services.* In other words, a LBAC system evaluating a policy does not have direct access to location information; rather, it sends location requests to external services and waits for the corresponding answers. The characteristics of these Location Services will depend on the communication environment where the user transaction takes place. Here, we focus on the mobile network, where Location Services are provided by mobile phone operators.

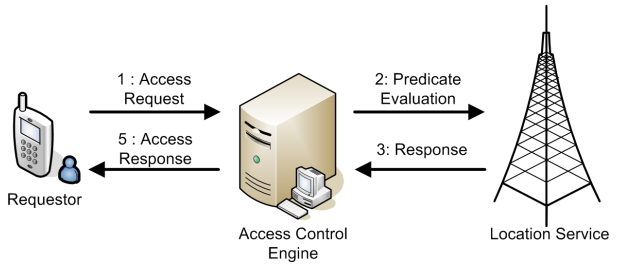

Figure 1: Location-based Architecture

Our LBAC architecture involves the following three entities.

Requestor. The entity whose access request to a service must be authorized by a LBAC system. We make no assumption about requestors, besides the fact that they carry terminals enabling authentication and some form of location verification.

Access Control Engine (ACE). The entity that implements the LBAC system used to enforce access control to the available services. To evaluate access requests according to some LBAC policies, ACE must communicate with a Location Service for acquiring location information.

Location Service (LS). The entity that provides the location information. The types of location requests that it can satisfy depend on the specific mobile technology, the methods applied for measuring users position, and environmental conditions.

Figure 1 depicts the architecture schema. Interactions among the Requestor, the Access Control Engine, and the Location Service is carried out via request/response message exchanges. The Access Control Engine receives access requests, evaluates policies and returns answers, invoking the Location Service when necessary. This functional decomposition is due to the fact that location functionalities are fully encapsulated within remote services set up and managed by the mobile operators. Therefore, no assumption can be made on these services besides their interfaces.

2.2 Location-based conditions

Intuitively, a *location-based condition* is a condition involving a predicate whose value depends on location measurements performed by a Location Service. Location-based predicates have been investigated since long by the wireless network research community [1], trying to address critical issues like time and space dependency. Two key issues are specific to LBAC:

- *interoperability*: location tracking can rely on different sources of location information, depending on availability and cost;
- *uncertainty*: each location measure, which a Location Service performs, has a degree of uncertainty due to technological limitations and possible environmental effects.

While the former issue largely depends on roaming agreements between mobile phone operators and is more business-oriented in nature, the latter needs to be tackled effectively for LBAC to reach its goals. Today, in the mobile network scenario, no technology is available ensuring fully exact user location [13]. The location *accuracy* is always less than 100%, so normally a position is specified as a range, locating the user within a certain area.[1] For a given location request, this area cannot be set *a priori*; rather, it may depend on the number of nearby antennas and on the surrounding landscape features. Also, a location measurement is often unstable because of changing environmental conditions, such as reflection or interferences that may corrupt the signal. In our model, we take into account these aspects by assuming that the result provided by a Location Service is always affected by a measurement error. This fact is relevant to the syntax and semantic of the Location Service interface because the outcome of the evaluation of an access request determined by the Access Control Engine will depend on such an uncertainty, which must then be explicitly represented and processed.

It is worth noting that performance-related properties of a localization service largely depend on the underlying technology. We use GSM/3G technologies as our reference due to their widespread usage and for recent advancements that have sensibly improved location capabilities.[2] Other technologies like 802.11 WiFi and AGPS/GPS [10, 22] could also be exploited although some limitations reduce their applicability. WiFi, for example, has a limited coverage and its usage is restricted to indoor environments (e.g., buildings, airports, malls) or in urban areas covered by many hotspots. GPS, on the contrary, does not work indoor or in narrow spaces but has no coverage limitation, a feature which makes it an ideal location technology for open, outdoor environments. The main techniques used in GSM/3G technology for location are the following.

Cell Identification. It is the simplest technique and is based on the identification of the mobile terminal serving cell. The coordinates of the cell provide a broad estimation of a user position, depending on the radius of the cell, which can be comprised between 200 meters and 2.5 kilometers. In towns, cells are much smaller than in the countryside.

Signal Level. It is based on measuring the signal attenuation. Assuming free space propagation and omnidirectional antennas, signal level contours around a base station are concentric circles where smaller circles enjoy more powerful signal. Directional antennas lead to more complex geometrical shapes. Unless advanced (and computationally heavy) ray-tracing algorithms are used, the signal level metric is not well-suited indoor or for urban areas.

Angle of Arrival (AoA). It assumes that several base stations are used for signal reception. A user position can be calculated by computing the angle of arrival at two base stations. Note, however, that if there is no line-of-sight between the mobile terminal and the base stations, the calculated angle do not correspond with the actual directional vector from the base station to the mobile.

Time of Arrival (ToA). The distance between a base station and a mobile phone is calculated by measuring the time a signal takes to make a round-trip between the two. Geometrically, this provides a circumference, centered at the base station. If more than three base stations are available, the intersection of their circles provides a mobile phone position. However, signal arrival can be delayed by walls or natural obstacles, decreasing location accuracy.

Time Difference of Arrival (TDoA). Assuming that base stations within the network are synchronized or propagation delays between them are known, the difference between station-to-terminal propagation times can be computed, increasing location accuracy. This can either be realized by measuring the differences between the arrival time of a certain burst sent by the mobile to several base stations or by recording the time differences of impinging signals at the mobile.

A positive aspect of all five location methods listed above is that they do not require any modification to existing mobile networks. Their accuracy and measurement error, however, may vary in different environments. In particular, TDoA and (enhanced) signal level methods are the most accurate in urban and indoor environments while AoA is well-suited for outdoor location because urban environments involve refraction and reflection phenomena that may compromise its performance.

3. LOCATION-BASED CONDITIONS AND PREDICATES

3.1 Expressing location-based conditions

A first step to support location-based conditions in an authorization language is to identify how location information is queried and what kind of response the Location Service returns. Traditional location-based services [16] usually assume queries to the Location Service to be of the form of *range queries* asking for an estimated range of values (possibly collapsing to a single value) for a predicate. Range queries can be modeled as functions of the form:

$$predicate(parameters) \rightarrow [range, accuracy, timeout]$$

stating that the evaluation of a location *predicate* over *parameters* returns a result of *range*. The *range* has a given *accuracy* that represents the radius of the circular area, where *range* is the center of such an area. Intuitively, *range* and *accuracy* represents the area where a terminal is located. The *accuracy* is therefore an upper bound of the deviation with respect to the true location of the terminal, guaranteed by the Location Service[3] and is to be considered valid within the timeframe specified by the *timeout*. For the sake of simplicity, in our model we consider queries to be of a simpler (although largely equivalent) form; namely, we shall

[1]Although elevation also counts, for the sake of simplicity we disregard it in this paper. Our results can be readily extended to incorporate 3D intervals, at the price of some additional complexity.

[2]In cooperation with Siemens Mobile, our group has recently patented a high-accuracy method for locating mobile phones suitable for indoor environments [2].

[3]Accuracy is a qualitative concept and should not be confused with *precision*, that is, the closeness of agreement between independent test results obtained under stipulated conditions.

deal with *boolean queries* asking the Location Service to assess whether a given value (or range thereof) of a location predicate is true or false.

Boolean queries can be modeled as functions of the form:

$$predicate(parameters,value) \rightarrow [bool\_value, confidence, timeout]$$

stating whether or not (depending on whether *bool_value* is `True` or `False`) *predicate* over *parameters* has the stated *value*. For instance, a query may ask whether a terminal is located inside a given region. Here, the assessment (`True` or `False`) has again a time validity specified by a *timeout* parameter; but instead of providing a measure accuracy, we assume that the Location Service attaches to answers a *confidence value*.

- The *confidence value* expresses the level of reliability that the Location Service is willing to guarantee to the assessment (`True` or `False`), according to accuracy, environmental and weather conditions, granularity of the requested location, and measurement technique. While confidence is associated with measurement *accuracy*, which in turn depends on the technology used for localizing the requester, the quality of the Location Service and so forth, the form of this dependency is encapsulated within the Location Service.

- The *timeout* represents the time validity of the result. This timeout takes into account that location values may change rapidly, even during policy evaluation. If the evaluation of a condition involving a predicate happens to start after the predicate timeout is expired, predicate re-evaluation is triggered.

Limiting ourselves to boolean queries simplifies the format of policy rules but does not represent a real constraint on expressive power. Intuitively, a range query with a condition on the returned range can be expressed as a boolean query where the condition is moved within the predicate itself. For instance, a condition requesting the area where the user is and then evaluating whether the area is `Milan`, can be represented as a condition requesting whether it is true or not that the user is in the `Milan` area.

Its worth noting that in range queries the Location Service can respond with different ranges and accuracy levels, thus varying the granularity of the response. For instance, the service can choose between a high-accuracy answer specifying a wide range (e.g., a city) or a low-accuracy answer specifying a smaller region (e.g., a building). In boolean queries, the accuracy is essentially established by the Access Control Engine, which sets the granularity at which the request is to be evaluated (e.g., asking the Location Service whether the user is located within a building or within a city, depending on the granularity needed for evaluating the access control policy). The Location Service answers stating whether the predicate is true or false, together with the confidence it has in such a response. The rationale behind our choice (i.e., boolean queries with a level of confidence) is to decouple the physical measurement error from the access control condition that the Access Control Engine has to evaluate. The Location Service is in the best position for providing a confidence estimate, because associating confidence with a range requires educated guesses about the measured variable probability distribution, as well as the knowledge of the number of physical measurements actually taken by the sensors. Our choice enables the Access Control Engine to evaluate location-based conditions without taking into account technological details of the location measurement process. An additional benefit of our approach is to foster interoperability between the Access Control Engine and multiple Location Services, possibly relying on different location technologies. Given a certain confidence in the evaluation of a location-based predicate (e.g., saying that a requester has been positively localized in a given area with a confidence level of 90%), the Access Control Engine will compute the final outcome of boolean location-based predicates by means of its local *confidence thresholds* (see Section 5). The confidence and timeout with which a Location Service responds to queries can be set via *Service Level Agreements* (SLAs) between the Access Control Engine and the Location Service.

3.2 Location-based predicates

The first step in the definition of location-based conditions is to identify the kind of conditions that it might be useful to include in access control policies and whose evaluation is possible with today's technology. We identified three main classes of conditions:

- *position-based* conditions on the location of the user; for instance, to evaluate whether a user is in a certain building or city or in the proximity of other entities;

- *movement-based* conditions on the mobility of the users, such as their velocity, acceleration, or direction where they are headed;

- *interaction-based* conditions relating multiple users or entities; for instance, the number of users within a given area.

We have also defined some specific predicates corresponding to specific conditions of the kind identified by the classes above. Our language is extensible with respect to the predicates and other predicates can be added as the need arises and technology progresses.

The language for location-based predicates assumes the following two elements.

- Users is the set of *user identifiers* (UID) that unambiguously identify users known to the Location Services. This includes both users of the system (i.e., potential requestors) as well as any other known physical and/or moving entity which may need to be located (e.g., a vehicle with an on-board GPRS card). A typical UID for location-based applications is the SIM number linking the user's identity to a mobile terminal.[4]

- Areas is a set of map regions identified either via a geometric model (i.e., a range in a n-dimensional coordinate space) or a symbolic model (i.e., with reference to entities of the real world such as cells, streets, cities, zip code, buildings, and so on) [17].

In the following, we will refer to elements of Users and of Areas as *user* and *area terms*, respectively. Note that, while

[4]Individual users may carry multiple SIMs and SIMs may be passed over to other users. We shall not elaborate on these issues, as strong identity management in mobile networks is outside the scope of this paper.

we assume such elements to be ground in the predicates, our language could be readily extended to support variables for them.

All our predicates are expressed as boolean queries, and therefore have the form *predicate*(*parameters, value*) as illustrated in Section 3.1. Their evaluation returns a triple [*bool_value, confidence, timeout*]. Whenever there is no risk of confusion, we will omit the predicates result.

Our core set of location predicates includes the following predicates (see Table 1).

- A binary *position* predicate inarea whose first argument is a user term and second argument is an area term. The predicate's semantics is evaluating whether a user is located within a specific area (e.g., a city, a street, a building).
- A binary *position* predicate disjoint whose first argument is a user term and second argument is an area term. The predicate's semantics is evaluating whether a user is outside a specific area. Intuitively, disjoint is the equivalent to the negation of inarea.
- A 4-ary *position* predicate distance whose first argument is a user term, second argument is either a user or area term (identifying an *entity* in the system), while the third and fourth arguments are two numbers specifying the minimum (*min_dist*) and maximum (*max_dist*) distance, respectively. The semantics of this predicate is to request whether the user lies within a given distance from the specified entity. The entity involved in the evaluation can be either stable or moving, physical or symbolic, and can also be the resource to which the user is requesting access. Note that exact distance can be evaluated by setting the same value for *min_dist* and *max_dist*, while "closer than" conditions can be evaluated by setting *min_dist* to 0. Finally, "farther than" conditions can be evaluated by setting *max_dist* to infinity.
- A ternary *movement* predicate velocity whose first argument is a user term, while the second and third arguments are two numbers specifying a minimum (*min_vel*) and maximum (*max_vel*) velocity, respectively. The semantics of the predicate is to request whether the user speed lies within a given range of velocity. Similarly to what happens for distance, exact velocity can be requested by setting the same value for *min_vel* and *max_vel*, while "smaller than" or "greater than" conditions can be evaluated by setting *min_vel* equal to 0 or a *max_vel* equal to infinity, respectively.
- A ternary *interaction* predicate density whose first argument is an area term, while second and third arguments are numbers specifying a minimum (*min_num*) and maximum (*max_num*) of users. The semantics of the predicate is to request whether the number of users currently in an *area* lies within the interval specified.
- A 4-ary *interaction* predicate local_density whose first argument is a user, the second argument is a "relative" area with respect to the user, while the third and fourth arguments specify a minimum (*min_num*) and maximum (*max_num*) of users, respectively. The semantics of the predicate is to evaluate the density within an area surrounding the user.

EXAMPLE 1. *Let* AliceSIM *be an element of* Users *and* Milan *and* Director Office *be two elements of* Areas *(specifying two symbolic characterizations corresponding to two known ranges of spatial coordinates).*

inarea(Alice,Milan) = [True,0.9,2005-11-09_11:10am]
states that the Location Service assesses as true the fact that Alice *is located in* Milan *with a confidence of* 90%*; and that such an assessment is to be considered valid until* 11:10am *of* November 9, 2005*.*

velocity(Alice,70,90) = [True,0.7,2005-11-03_03:00pm]
states that the Location Service assesses as true the fact that Alice *is traveling at a speed included in the range* [70,90] *with a confidence of* 70%*, and that such an assessment is to be considered valid until* 3:03pm *of* November 3, 2005*.*

density(Director Office,0,1) = [False,0.95,2005-11-21_06:00pm]
states that the Location Service assesses as false the statement that there is at most one person in the Director Office *and believes that two or more persons are in the office with a confidence of* 95%*. Such an assessment is to be considered valid until* 06:00pm *of* November 21, 2005*.*

4. LOCATION-BASED ACCESS CONTROL POLICIES

We now discuss how location-based access control policies can be expressed. Note that we will not attempt to develop a new language for specifying access control policies. Instead, our proposal can be thought of as a general solution for enriching the expressive power of existing languages (e.g., [6, 15, 21, 26]), by exploiting location information, without increasing the computational complexity of their evaluation. Here, we keep the language and the context representation as simple as possible, assuming that the Access Control Engine recognizes only users registered at the server. Each user is assigned an identifier. Besides their identifiers, registered users usually have other properties such as name, address, and date of birth. To capture and reason about these properties, we assume that each user is associated with a *user profile*. Objects are the data/services on which users can make requests. Abstractions can also be defined within the domain of objects, allowing to group together objects with common characteristics and to refer to the whole group with a single name. Similarly to subjects, objects have set of properties represented by means of an *object profile*. For the sake of simplicity and generality of the approach, we assume access control rules to be triples whose elements are generic boolean formula over the subject requesting access, the object to which access is requested, and the action the subjects wants to perform on it. Considering boolean formula over generic predicates and/or properties makes our approach applicable to (and compliant with) various proposals existing in the literature. Formally, an access control rule is defined as follows.

DEFINITION 1 (ACCESS CONTROL RULE). *An access control rule is a triple of the form* ⟨subj_expr, obj_expr, action⟩*, where:*

- subj_expr *is a boolean formula of terms that allows referring to a set of subjects depending on whether they*

Type	**Predicate**	**Description**
Position	inarea(*user*, *area*)	Evaluate whether *user* is located within *area*.
	disjoint(*user*, *area*)	Evaluate whether *user* is outside *area*.
	distance(*user*, *entity*, *min_dist*, *max_dist*)	Evaluate whether the distance between *user* and *entity* is within interval [*min_dist*, *max_dist*].
Movement	velocity(*user*, *min_vel*, *max_vel*)	Evaluate whether *user*'s speed falls within range [*min_vel*, *max_vel*].
Interaction	density(*area*, *min_num*, *max_num*)	Evaluate whether the number of users currently in *area* falls within interval [*min_num*, *max_num*].
	local_density(*user*, *area*, *min_num*, *max_num*)	Evaluate the density within a 'relative' area surrounding *user*.

Table 1: Examples of location-based predicates

satisfy or not certain conditions, where conditions can evaluate the user's profile, location predicates, or the user's membership in groups, active roles, and so on;

- obj_expr *is a boolean formula of terms that allows referring to a set of objects depending on whether they satisfy or not certain conditions, where conditions evaluate membership of the object in categories, values of properties on metadata, and so on;*
- action *is the action (or class of actions) to which the rule refers.*

We assume profiles to be referenced with the identity of the corresponding users/objects. Single properties within users and objects profiles are referenced with the traditional dot notation. For instance, `Alice.Address` indicates the address of user `Alice`. Here, `Alice` is the identity of the user (and therefore the identifier for the corresponding profile), and `Address` is the name of the property. To make it possible to refer to the user and object of the request being evaluated without need of introducing variables in the language, we introduce the following keywords.

- **user**. It indicates the identifier of the person making the request.
- **sim**. It indicates the SIM card number of the person making the request.
- **object**. It indicates the identifier of the object to which access is requested.

For instance, **user**`.Affiliation` indicates the property `Affiliation` within the profile of the user whose request is being processed.

Conditions specified in the *subj_expr* field can be classified into two categories: *generic conditions* and *location-based conditions*. Generic conditions evaluate membership of subjects into classes or properties in their profiles. For simplicity, we can assume that information stored at a service provider is sufficient to evaluate generic conditions. Cases where the Access Control Engine does not have any a priori knowledge of the user can be solved by assuming a negotiation/communication process between the two, eventually yielding to a state where the Access Control Engine does have all the information it needs (or it decides to deny the access). In this paper, we do not consider this phase of evaluation; rather, we refer to the many proposals existing in the literature that can be used for this step (e.g., [6, 28]). Our proposal can be seamlessly integrated with any of them. Location-based conditions are expressed using the location predicates described in Section 3.

In the following, we use $\mathcal{P}$ to denote the set of access control policies stored at the Access Control Engine. Given an access control rule $r \in \mathcal{P}$, *subj_expr*(r), *obj_expr*(r), *action*(r) will denote the subject expression, object expression, and action, respectively, of r.

EXAMPLE 2. *An important scenario for the application of location-based access control policies is accessing highly sensitive services, whose security requirements need both strong authentication methods and powerful and expressive access control policies. As an example of such critical services, let us consider a company which is responsible for the management of a mobile network. Managing a nation-wide mobile network is an extremely critical activity because reconfiguration privileges must be granted to strictly selected personnel only and must be performed according to the highest security standards. In addition to reconfiguration privileges, the access to logging and billing data is critical too, because they include information about the position and movements of mobile operator's customers. We consider access to customer-related information as less critical than reconfiguration privileges but still to be handled in a highly secured environment and to be granted only to selected personnel, according to the laws and regulations in force. Finally, at a lower criticality level, we consider access to statistical data about the network's operation. These data are however not public and must be protected, for example, from disclosure to competitors. We assume that the* Mobile Network Console *(MNC) is a software service that permits to reconfigure the mobile network and read customer data. We now present some examples of protection requirements for such a service; Table 2 reports the corresponding access control rules.*

1. *System administrators (*__user__`.Role=Admin`*), with a valid account, are authorized to configure the mobile network if they are in the server farm room (*inarea(**sim**, `Server Farm Room`)*), they are alone in such an area (*density(**sim**, `Server Farm Room`, `1`, `1`)*) and move at walking speed at most (*velocity(**sim**, `0`, `3`)*).*
2. *System administrators (*__user__`.Role=Admin`*), with a valid account, are authorized to read mobile network data if they are in the information systems dept. area*

	subject		action	object
	generic conditions	location conditions		
1	**user**.Role=Admin $\wedge$ Valid(**user**.Username, **user**.Password)	inarea(**sim**, Server Room) $\wedge$ density(Server Room, 1, 1) $\wedge$ velocity(**sim**, 0, 3)	Configure	MNC
2	**user**.Role=Admin $\wedge$ Valid(**user**.Username, **user**.Password)	inarea(**sim**, Inf. System Dept.) $\wedge$ velocity(**sim**, 0, 3) local_density(**sim**, Close By, 1, 1) $\wedge$	Read_Data	MNC
3	**user**.Role=CEO $\wedge$ Valid(**user**.Username, **user**.Password)	local_density(**sim**, Close By, 1, 1) $\wedge$ inarea(**sim**, Corporate Main Office) $\wedge$ velocity(**sim**, 0, 3)	Read_Data	MNC
4	**user**.Role=CEO $\wedge$ Valid(**user**.Username, **user**.Password)	local_density(**sim**, Close By, 1, 1) $\wedge$ disjoint(**sim**, Competitor Location)	Read_Statistics	MNC
5	**user**.Role=Guest $\wedge$ Valid(**user**.Username, **user**.Password)	local_density(**sim**, Close By, 1, 1) $\wedge$ inarea(**sim**, Corporate Location)	Read_Statistics	MNC

Table 2: Examples of access control rules regulating access to a Mobile Network Console

*(*inarea(**sim**, Information Systems Dept.)*), move at walking speed at most and there is nobody close by (*local_density(**sim**, Close By, 1, 1)*).*

3. *The CEO (***user**.Role=CEO*), showing a valid account, is authorized to access mobile network data if she is alone (there is nobody close by), she is in the corporate main office (*inarea(**sim**, Corporate Main Office)*), and moves at walking speed at most.*

4. *The CEO (***user**.Role=CEO*), showing a valid account, is authorized to access mobile network statistics if there is nobody close by and she is not in a competitor location (*disjoint(**sim**, Competitor Location)*).*

5. *Guests (***user**.Role=Guest*), with a valid account, can read mobile network statistics if there is nobody close by and they are in a corporate location (*inarea(**sim**, Corporate Location)*).*

5. POLICY EVALUATION AND ENFORCEMENT

We are now ready to discuss how access control policies enriched with location-based conditions are evaluated.

5.1 From confidence to truth values

Before illustrating how the access control process operates, we need to solve a basic problem: location-based predicates appear in rules as parts of a boolean formula, while the responses to boolean location queries are in the form of a triple [*bool_value, confidence, timeout*]. Then to process a response from the Location Service, the Access Control Engine will need to assign a truth value to it.[5] Intuitively, the transformation of a location predicate's value into a boolean one requires the Access Control Engine to determine whether or not the value returned by the Location Service can be considered valid for the purpose of controlling access. Such an evaluation will depend on parameters *timeout* and *confidence* returned by the Location Service.

Indeed, responses with a timeout that has already expired automatically trigger the re-evaluation of the predicate regardless of the other parameter values because considered as unreliable for any decision. Responses with a timeout that has yet not expired are evaluated with respect to the confidence value.

[5]We note in passing that alternative solutions are possible, for example defining a fuzzy or probabilistic reasoning on the rules.

Predicate	Confidence Thresholds		MaxTries
	lower	upper	
inarea	0.1	0.9	10
disjoint	0.1	0.9	10
distance	0.2	0.8	5
velocity	0.2	0.8	5
density	0.3	0.7	3
local_density	0.3	0.7	3

Table 3: An example of Extended True Table for location predicates

The confidence value is compared with two thresholds, specified for each location predicate. According to the result of this comparison (i.e., whether the confidence value is greater than the upper threshold, less than the lower threshold, or between the two), the boolean value contained in the response to a boolean query will be treated differently.

Before proceeding further it is important to remark that, as anticipated in Section 3, our confidence value has a semantics of *belief*; namely a response returning a boolean value v with a confidence of α is to be considered equivalent to a response returning $\neg v$ with a confidence of $1 - \alpha$. Another important observation is that the threshold of confidence above which the Access Control Engine will consider as valid the value returned by the Location Service may vary depending on different predicates (since more or less certain information might be required) as well as on how much the Access Control Engine trusts the confidence assessment of the Location Service (e.g., a 80% confidence stated by a very reliable Location Service could be considered almost as a true value, while a 90% confidence stated by another - less reliable - Location Service can be considered as not reliable).[6] Table 3 illustrates an example of an Extended True Table (ETT) featuring custom *Confidence Thresholds* for each predicate.

The ETT is used as follows: if the *confidence* level for a given predicate is greater than the preset upper threshold

[6]This behavior is similar to reputation-based approaches developed for peer to peer environments, where peers' votes are weighted with respect to the credibility of the voters [7].

(column *upper* of the ETT table), then the boolean value returned by the Location Service will be confirmed. If the *confidence* level is below the lower threshold (column *lower* of the ETT table), the boolean value returned is not confirmed and the location-based condition is evaluated to $\neg$*bool_value*. Otherwise, if the *confidence* level is between lower and upper threshold neither the returned value nor its negation can be considered sufficiently reliable to take a decision about the location-based condition and predicate re-evaluation is triggered. The same happens when the *timeout* has expired prior to evaluation. To avoid deadlock, a *MaxTries* number is defined for each location predicate (column *MaxTries* in the ETT table), expressing the max number of predicate re-evaluations that the Access Control Engine will request either because the confidence level is not high enough or due to a timeout. If after *MaxTries* re-evaluations of the predicate, the outcome remains unchanged, the evaluation process ends and the final response is Undefined. Predicates whose evaluation is more time consuming are likely to be re-evaluated fewer times than others. In our case, density and local_density are the most time consuming predicates because their evaluation depends on the position of multiple entities. On the other hand, inarea and disjointare the least time consuming predicates since they depend on the position of a single entity; distance and velocity depend on the location measure of two entities.

With respect to the values set for *Confidence Thresholds* in Table 3, the rationale is that since they represent SLA defined terms, they should reflect the overall reliability of a location measurement provided by a specific Location Service[7]. Accordingly, thresholds are empirical values that an expert should estimate when SLA agreements are set. They should encapsulate both an evaluation of the technical difficulty of providing the measurement required by the specific location predicate and the Location Service reliability. Examples of technical aspects that may influence the confidence are the sensitivity of the predicate to external conditions, such as environmental and weather conditions, and measurement techniques adopted. Examples of factors that may affect a Location Service reliability are the expertise with the specific measurement techniques and the distribution and number of sensors of the Location Service infrastructure. According to this rationale, inarea and disjoint are the predicates that most suffer from environmental changes and consequently we should set a small confidence interval for reducing the uncertainty. Predicate density and local_density are the less sensitive and we may accept larger confidence intervals to confirm the result returned by the Location Service. Finally, predicates distance and velocity are in the middle with respect to confidence too. Actual values of Table 3 represent educated guesses, without, of course, considering the reliability of a specific Location Service.

To perform the mapping of boolean queries responses into boolean values we define a function Solve that enforces the semantics just described. The function, illustrated in Figure 2, takes as input a predicate name *pred-name* with its parameters $p_1, \ldots, p_n$, and a Location Service *LS* to be queried, and returns as output a value in the set {True, False, Undefined}.

[7] *Confidence Thresholds* could be set by considering how many re-evaluation attempts are available before the timeout expires. Predicates with more re-evaluation attempts have more possibilities to receive a response with a sufficient confidence level and could be set to higher thresholds than others.

```
Function Solve(pred-name($p_1, \ldots, p_n$),LS):
{True, False, Undefined}
    upper:=ETT[pred-name, upper];
    lower:=ETT[pred-name, lower];
    maxtries:=ETT[pred-name, MaxTries];
    response:= Undefined;
    tries := 0;
    repeat
        Send query pred-name($p_1, \ldots, p_n$) to LS
        Receive Reply = [bool_value, confidence, timeout]
        if current-time < timeout then
            case confidence of
                $\geq$ upper: response := bool_value;
                $\leq$ lower: response := $\neg$ bool_value;
        endif
        tries := tries + 1;
    until (response$\neq$Undefined) or (tries > maxtries)
    return response
```

Figure 2: Function Solve

EXAMPLE 3. *Let* Alice-sim *be an element of* Users *and* LS *be the Location Service associated with it. Suppose that the* ETT *is as illustrated in Table 3 and that the current time is* 10:45AM *of* Nov. 9, 2005. *Consider then the following calls of function* **Solve**.

- **Solve**(inarea(Alice-sim,Inf. System Dept.),LS)

  Suppose that inarea(Alice-sim,Inf. System Dept.) = [True,0.95,2005-11-09_11:00AM], *the function returns* True.

- **Solve**(velocity(Alice-sim,0,3),LS)

  Suppose that velocity(Alice-sim,0,3) = [True,0.9, 2005-11-09_10:50AM], *the function returns* True.

- **Solve**(local_density(Alice-sim,Close By,1,1),LS)

  Suppose that a first query evaluates local_density(Alice-sim,Close By, 1,1) = [True,0.6, 2005-11-09_11:10AM]. *Since the confidence falls within the uncertain range the query is submitted a second time.*

  Suppose that the second attempt returns [True,0.65,2005-11-09_11:12AM]. *Again, since the confidence falls within the uncertain rang, a third attempt is performed.*

  Suppose that the third attempt returns [True,0.63,2005-11-09_11:13AM]. *Again, the confidence falls within the uncertain range. Since* maxtries *has been reached, no further query is requested and the function returns* Undefined.

5.2 Access control enforcement

We are now ready to describe how the access control process operates. We start by characterizing the access requests submitted to the Access Control Engine.

Definition 2 (Access request). *An* access request *is a 4-tuple of the form* ⟨user_id, SIM, action, object_id⟩, *where* user_id *is the optional identifier of the user who makes the request,* SIM $\in$ Users *is the optional SIM card number,* action *is the action that is being requested, and* object_id *is the identifier of the object on which the user wishes to perform the action.*

We assume that the Access Control Engine evaluates first whether a decision can be taken locally (i.e., based on rules evaluating only generic conditions). If no decision can be taken locally (all applicable authorizations involve location-based predicates), the corresponding queries are sent to the involved Location Service. The reason for such an assumption is that location-based predicate evaluation usually bears cost and therefore is to be avoided whenever possible. More precisely, the policy evaluation and enforcement process (involving the communications illustrated in Figure 1) can be described as a three-phase process as follows.

Let ⟨*user_id*, *SIM*, *action*, *object_id*⟩ be an access request. In the first phase, the Access Control Engine evaluates the policy $\mathcal{P}$ collecting all the rules $\mathcal{A} \in \mathcal{P}$ that are applicable to the request. The set $\mathcal{A}$ of applicable rules contains those rules $r \in \mathcal{P}$ for which *action*(r) corresponds to the *action* specified in the access request, and *object_id* satisfies the conditions specified in *obj_expr*(r). Let $\mathcal{A}_g \subseteq \mathcal{A}$ be the set of applicable rules, where the subject expressions contain generic conditions only, and $\mathcal{A}_{lp} \subseteq \mathcal{A}$ be the set of applicable rules where the subject expressions contain location-based predicates. If there exists a rule $r \in \mathcal{A}_g$ such that *subj_expr*(r) evaluates to true, then the access is granted and the evaluation process ends. For instance, suppose that the subject expression of an applicable rule $r \in \mathcal{A}_g$ is of the form (**user**.Company=ACME $\wedge$ **user**.Job=employee) and the requestor is an employee. In this case, the subject expression is evaluated to true and the access is granted.

In the second phase, for each rule $r \in \mathcal{A}_{lp}$, the Access Control Engine simplifies *subj_expr*(r) applying the usual rules of propositional calculus. More precisely, subject expression *subj_expr*(r) is first simplified by evaluating all the generic conditions. For instance, suppose that the subject expression of an applicable rule $r \in \mathcal{A}_{lp}$ is of the form (**user**.Citizenship=EU $\wedge$ inarea(**sim**, Venice)) and the requestor is Italian. In this case, the residual subject expression is (True $\wedge$ inarea(**sim**,Venice)), that is, inarea(**sim**,Venice). This condition, called *residual condition*, cannot be further simplified and the access request cannot be immediately granted or denied. Consequently, the Access Control Engine proceeds evaluating location-based conditions. More precisely, for each predicate *pred-name*($p_1, \ldots, p_n$), the Access Control Engine determines the Location Service *LS* involved and calls function **Solve**(*pred-name*($p_1, \ldots, p_n$),*LS*).[8]

In the third phase, all conditions' outcomes are put together to reach the final access control decision. There is a slight complication here: the resolution of predicates in the subject expression can have three possible values, namely True, False, Undefined. The boolean expression outcome then results from the application of the classical truth tables of propositional connectives $\vee$, $\neg$, and $\wedge$ defined in the 3-valued logic. In particular: Undefined $\wedge$ False=False, Undefined $\vee$ True=True. Other disjunctions or conjunctions involving Undefined, as well as the negation of Undefined itself, have as result Undefined.

Access is granted, if for an applicable rule, the subject expression evaluates to True; it is denied otherwise.

[8]We can imagine that, since location queries bear some cost, the Access Control Engine will request location queries considering one authorization at the time (stopping the process when an authorization is satisfied).

Example 4. *Let* Alice *be a user and* Alice-sim *her SIM's number. Assume now that* Alice *requests* Read_data *access to the* MNC *(Mobile Network Console). Let* $\mathcal{P}$ *be the set of policies described in Table 2.*

The access control proceeds as follows. First, the set of applicable rules $\mathcal{A}$ *is retrieved. This is the set containing authorizations 2 and 3, both of which fall in* $\mathcal{A}_{lp}$*. Then, the authorizations are simplified to retrieve residual (location-based) conditions.*

Suppose Alice*'s active role is* Admin *and that* Alice *is connected with a valid account. Consequently, the generic conditions in authorization 3 evaluate to* False*; making the whole authorization evaluate to* False*. In contrast, the generic conditions in authorization 2 evaluate to* True*, resulting in the residual condition to be evaluated as: "*local_density(Alice-sim,Close By,1,1) $\wedge$ velocity(Alice-sim,0,3) $\wedge$ inarea(Alice-sim, Inf. System Dept.)*"*

For each of the predicates in the residual condition, function **Solve** *is called. Assume function* **Solve** *to follow the process illustrated in Example 3, producing:*

- **Solve**(local_density(Alice-sim, Close By, 1, 1), LS) = Undefined
- **Solve**(inarea(Alice-sim, Inf. System Dept.), LS) = True
- **Solve**(velocity(Alice-sim, 0, 3), LS) = True

Hence, Undefined $\wedge$ True $\wedge$ True = Undefined *and the access is denied.*

6. RELATED WORK

The diffusion and reliability that mobile technologies have reached provide a means to exploit location information for improving current access control systems in a novel way. To this end, the definition of a LBAC model is an emerging research issue that has not been yet addressed by the security and access control research community. The notion of location-based access control is however in itself not new. Some early mobile networking protocols already incorporated the notion of linking the physical position of a terminal device and its capability of accessing network resources [1]. Widespread adoption of wireless local networks has triggered new interest in this topic. Some recent studies focused on location-based information for monitoring users movements on Wireless Lan [8] and 802.11 Networks [9], while Myllymaki and Edlund [19] described a methodology for aggregating location data from multiple sources and improve this way location tracking features. Other researchers chose a line closer to our approach by describing the architecture and operation of an access server module for LBAC in local wireless networks [20, 27]. Access control to wireless networks complying with IEEE 802.11 family of protocols is currently being standardized. However, these contributions were aimed at strengthen the well-known security weaknesses of wireless network protocol rather than defining a general, protocol-independent model for LBAC. The need for a protocol-independent location technique has been underlined by an interesting study exploiting heterogeneous positioning sources like GPS, Bluetooth, and WaveLAN for designing location aware application [20]. Location-based information and their management are also the topics of a recent study by Varshney [27] in the area of mobile commerce applications. This is a related research area that has strong connection with location systems and is a promising source of requirements for LBAC models, for example, when particular purchase options or taxation exemptions must be applied to customers of a specific region or country.

Few papers, instead, consider location information as a means for improving security. Sastry et al. [24] exploit location-based access control in sensor networks. Zhang and Parashar [29] proposed a location-aware extension to Role-Based Access Control (RBAC) suitable for grid-based distributed applications. Many papers take into account time variant information for querying database containing location information [14, 18]. Typically, in such approaches a user location is treated as a single point without explicitly consider the intrinsic uncertainty of each location measurement.

Other papers present some methods to track a user in motion. These methods are based over a user habits profile and, again, consider the users as a single point. For instance, in [23], a framework for user mobility prediction is presented. It can predict the traveling trajectory and destination using knowledge of user's preferences, goals, and analyzed spatial information without imposing any assumptions about the availability of users' movements history. This framework thus incorporates the notion of combining user context and spatial conceptual maps in the prediction process. All these contributions, however, do not address the uncertainty intrinsic in location information.

Other works took a different approach with respect to location information as resources to protect against unauthorized access. For instance, in [12], a mechanism to protect a user's location information by means of electronic certificate negotiation, delegation and trusted location-based services is described. The same problem is addressed in [11] by proposing a privacy-aware architecture for a *global Location Service*, which should allow users to define rules that will be evaluated to manage accesses to location information. *Mix zones* is the method developed in [4] to enhance privacy in location-based services that make use of a trusted middleware service. Some basic concepts for the definition of privacy policies in location-based environments are discussed in [25], while [5] presents a preliminary investigation of the privacy issues raised by location-based services. Finally, the idea of enabling users to write their own privacy policies to control the disclosure of their location information is developed in [3]. Such feature will be integrated in our LBAC model.

7. CONCLUSIONS

In this paper, we have presented a model for representing and evaluating LBAC conditions. Our approach encapsulates time-dependency and uncertainty of location measurements as important features of location information in a small set of semantically uniform service level agreement (SLA) parameters based on the notions of confidence level and temporal validity of each access request. These parameters are aimed at achieving a clean separation of the access control evaluation engine from the protocol-dependent Location Services made available by the network operators. Using policy conditions based on these shared parameters rather than on protocol-specific entities will ensure consistent access decisions regardless of the available location technology and of the environment's transient conditions. Formal definitions of a number of location-based predicates have been provided, together with a discussion of their management, evaluation and enforcement. Also, we proposed an architecture to integrate our LBAC evaluation with traditional identity-based access control. Our proposed architecture can support a broad variety of location-based policies and a rich set of predicates. Finally, we presented a worked-out example of a LBAC policy applied to a relevant industrial scenario where new security requirements can be met by exploiting our location-based predicates. Several open issues still remain. A particularly interesting issue concerns the specification and enforcement of security (and privacy) constraints on location-based information. In particular, in this paper we have assumed that the Location Service always returns location information. We plan to extend our approach to the consideration of policies constraining to whom and how location information is to be provided. Such policies can come, for example, from user's privacy preferences or legislations.

8. ACKNOWLEDGMENTS

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591 and by the Italian MIUR within the KIWI and MAPS projects.

9. REFERENCES

[1] I.F. Akyildiz and J.S.M. Ho. Dynamic mobile user location update for wireless pcs networks. *Wireless Networks*, 1(2):187–196, 1995.

[2] M. Anisetti, C.A. Ardagna, V. Bellandi, and E. Damiani. Positioning method and system for mobile communications networks, related networks and computer program product. European Patent No. 05425643.3, Deposited in date 15 September 2005.

[3] C.A. Ardagna, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Towards privacy-enhanced authorization policies and languages. In *Proc. of the 19th IFIP WG11.3 Working Conference on Data and Application Security*, Nathan Hale Inn, University of Connecticut, Storrs, USA, August 7-10 2005.

[4] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *Proc. of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW04)*, Orlando, Florida, March 2004.

[5] C. Bettini, X.S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification.

In *Proc. of the 2nd VLDB Workshop on Secure Data Management*, Trondheim, Norway, September 2005.

[6] P. Bonatti and P. Samarati. A unified framework for regulating access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.

[7] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servents' reputations in p2p systems. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):840–854, July/August 2003.

[8] D. Faria and D. Cheriton. No long-term secrets: Location-based security in overprovisioned wireless lans. In *Proc. of the Third ACM Workshop on Hot Topics in Networks (HotNets-III)*, San Diego, USA, November 2004.

[9] S. Garg, M. Kappes, and M. Mani. Wireless access server for quality of service and location based access control in 802.11 networks. In *Proc. of the Seventh IEEE Symposium on Computers and Communications (ISCC 2002)*, Taormina/Giardini Naxos, Italy, July 2002.

[10] I. Getting. The global positioning system. *IEEE Spectrum*, 30(12):36–47, December 1993.

[11] C. Hauser and M. Kabatnik. Towards Privacy Support in a Global Location Service. In *Proc. of the IFIP Workshop on IP and ATM Traffic Management (WATM/EUNICE 2001)*, Paris, France, 2001.

[12] U. Hengartner and P. Steenkiste. Implementing access control to people location information. In *Proc. of the ACM Symposium on Access Control Models and Technologies 2004 (SACMAT 2004)*, Yorktown Heights, USA, 2004.

[13] S. Horsmanheimo, H. Jormakka, and J. Lahteenmaki. Location-aided planning in mobile network trial results. *Wireless Personal Communications: An International Journal*, 30(2-4):207–216, September 2004.

[14] H. Hu and D.L. Lee. Energy-efficient monitoring of spatial predicates over moving objects. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 28(3):19–26, 2005.

[15] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2):214–260, June 2001.

[16] U. Leonhardt and J. Magee. Towards a general location service for mobile environments. In *Proc. of the 3rd Workshop on Services in Distributed and Networked Environments (SDNE'96)*, Macau, June 1996.

[17] N. Marsit, A. Hameurlain, Z. Mammeri, and F. Morvan. Query processing in mobile environments: a survey and open problems. In *Proc. of the First International Conference on Distributed Framework for Multimedia Applications (DFMA'05)*, Besancon, France, February 2005.

[18] M.F. Mokbel and W.G. Aref. GPAC: Generic and progressive processing of mobile queries over mobile data. In *Proc. of the 6th international conference on Mobile data management*, Ayia Napa, Cyprus, May 2005.

[19] J. Myllymaki and S. Edlund. Location aggregation from multiple sources. In *Proc. of the 3rd IEEE Int.l Conf. on Mobile Data Management (MDM 02)*, January 2002.

[20] J. Nord, K. Synnes, and P. Parnes. An architecture for location aware applications. In *Proc. of the 35th Hawaii Int.l Conference on System Sciences*, Hawaii, USA, 2002.

[21] OASIS. *eXtensible Access Control Markup Language (XACML) Version 1.0*, 2003. http://www.oasis-open.org/committees/xacml.

[22] B. Parkinson, J. Spilker, P. Axelrad, and P. Enge, editors. *Global Positioning System: Theory and Application, Volume II.* American Institute of Astronautics and Aeronautics (AIAA), 1996.

[23] N. Samaan and A. Karmouch. A mobility prediction architecture based on contextual knowledge and spatial conceptual maps. *IEEE Transaction on Mobile Computing*, 4(6):537–551, November-December 2005.

[24] N. Sastry, U. Shankar, and S. Wagner. Secure verification of location claims. In *Proc. of the ACM Workshop on Wireless Security (WiSe 2003)*, San Diego, CA, USA, September 2003.

[25] E. Snekkenes. Concepts for personal location privacy policies. In *Proc. of the 3rd ACM conference on Electronic Commerce*, Tampa, Florida, USA, 2001.

[26] T.W. van der Horst, T. Sundelin, K.E. Seamons, and C.D. Knutson. Mobile trust negotiation: Authentication and authorization in dynamic mobile networks. In *Proc. of the Eighth IFIP Conference on Communications and Multimedia Security*, Lake Windermere, England, September 2004.

[27] U. Varshney. Location management for mobile commerce applications in wireless internet environment. *ACM Transactions on Internet Technology*, 3(3):236–255, August 2003.

[28] T. Yu, M. Winslett, and K.E. Seamons. Supporting structured credentials and sensitive policies trough interoperable strategies for automated trust. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):1–42, February 2003.

[29] G. Zhang and M. Parashar. Dynamic context-aware access control for grid applications. In *Proc. of the 4th International Workshop on Grid Computing (Grid 2003)*, Phoenix, Arizona, November 2003.