A Formal Proximity Model for RBAC Systems

Aditi Gupta
Department of Computer Science
Purdue University
aditi@purdue.edu

Michael Kirkpatrick
Department of Computer Science
James Madison University
kirkpams@jmu.edu

Elisa Bertino
Department of Computer Science
Purdue University
bertino@purdue.edu

***Abstract*—To combat the threat of information leakage through pervasive access, researchers have proposed several extensions to the popular role-based access control (RBAC) model. Such extensions can incorporate contextual features, such as location, into the policy decision in an attempt to restrict access to trustworthy settings. In many cases, though, such extensions fail to reflect the true threat, which is the presence or absence of *other users*, rather than absolute locations. For instance, for location-aware separation of duty, it is more important to ensure that two people are in the same room, rather than in a designated, pre-defined location. Prox-RBAC was proposed as an extension to consider the relative *proximity* of other users with the help of a pervasive monitoring infrastructure. However, that work offered only an informal view of proximity, and unnecessarily restricted the domain to spatial concerns. In this work, we present a more rigorous definition of proximity based on formal topological relations. In addition, we show that this definition can be applied to several additional domains, such as social networks, communication channels, attributes, and time; thus, our policy model and language is more flexible and powerful than the previous work. In addition to proposing the model, we present a number of theoretical results for such systems, including a complexity analysis, templates for cryptographic protocols, and proofs of security features.**

I. Introduction

The rise of mobile and pervasive computing has made it possible to devise context-aware systems that customize the computing experience to the user's environment. One particular application for these systems is to facilitate the design of access control systems that aim to mitigate the threat of data loss by restricting permissions to appropriate settings. As these concerns are more relevant to enterprise settings, researchers often use RBAC as the foundation for designing such access control models and systems. For instance, several models have been proposed that consider the requesting user's location in the policy decision [8], [1], [2].

While such extensions to RBAC can provide a basis for reasoning about contextual policies, they fail to reflect many of the more interesting scenarios. Specifically, it may be more important to consider the relative locations of *other users*, rather than the requesting user's location. For instance, when preparing a financial deposit slip in a retail setting, the presence of a supervisor may be critical, while ensuring that the employee is in the store office is not. To enable the creation of such policies, Prox-RBAC [17] was proposed to incorporate *proximity constraints* into a spatial RBAC model. That is, Prox-RBAC policies consisted of a spatial RBAC policy with an additional clause specifying constraints on the locations of other users; for instance, one can specify a constraint for a military deployment that no civilians be present.

We have identified two shortcomings with Prox-RBAC as previously proposed. First, the model relies on an intuitive notion that "proximity" means the users are present (or not) within the same physical space. This lack of a rigorous understanding of proximity can lead to surprising interpretations. For instance, in Prox-RBAC, two users at opposite ends of a building could be considered to be within proximity for one policy; however, for another policy, two users standing in adjacent rooms on opposite sides of the same door would *not* be in proximity of one another. As such, this informal approach allows for entities to be in proximity, despite the fact that they are not physically close.

Second, we find the exclusive focus on the spatial domain to be unnecessarily restrictive. The intuition that proximity indicates relative closeness of two entities can be applied in several domains with interesting results. For instance, a temporal proximity restraint could require that two people digitally sign a document within 24 hours. In attribute-based proximity, an Assistant Professor and an Associate Professor have professions (*i.e.,* attributes) that are similar. Clearly, a unified and formal definition of proximity can be applied to a wide variety of settings.

We have analyzed five contextual domains, or *realms*, namely geographic, attribute-based, social, cyber, and temporal realms for defining proximity. We will start this work by defining these realms and showing how they can be mapped onto a unified abstract space model. We will then apply the calculus-based method [5] for defining topological relations on *features* in order to specify a formal distance metric. We then use this metric to define two forms of proximity, specifically *weak role proximity* and *strong role proximity*. In both forms, proximity specifies that two entities must have a distance measure (in the abstract space) that is less than some threshold value.

In addition to defining the model, we also present a number of theoretical results and practical advice for the creation of proximity-based RBAC systems. We show that, given a single policy with proximity constraints, determining whether or not it can be satisfied is NP-hard. Furthermore, if the mapping of users to features can be done in polynomial time, then the problem is NP-complete. We also show that correct evaluation of proximity constraints is impossible if the deployment allows for asynchronous communication.

However, despite these results, practical deployments are still feasible. Specifically, the NP-hard result depends on the complexity of the proximity constraints; if only simple proximity constraints are used, evaluating the policy design becomes tractable. The impossibility result, on the other hand, applies to the simultaneous evaluation of the constraint for a request while another user changes features. That is, there is a small window of time when the policy information point (PIP) has an inconsistent view of the system. Note, though, that this is only relevant if the user who is changing features has some impact on the proximity constraint under consideration; if that user is not relevant to the current policy, this inconsistency has no effect on the policy decision. One can further mitigate this threat of inconsistency with redundancy; that is, by repeating the constraint evaluation after a small window of time. Thus, while perfect guarantees are impossible, the system may be able to enforce the policies well enough for practical concerns.

It is important to emphasize the advantages of this formal approach. First, by grounding the notion of proximity in terms of a distance and threshold values, we ensure that our formalisms reflect the intuition of proximity as closeness. As such, the mandatory specification of a metric reduces the likelihood of surprising interpretations of proximity. Second, by defining proximity in terms of an abstract space model, our approach is very flexible and simplifies the adaptation of policies for other realms beyond the five we consider. That is, mapping the realms onto the abstract space model allows us to define a common framework for enforcing the policy constraints; adapting the model and policies for additional realms would only require mapping the realm onto the abstract space model. Finally, by defining a common enforcement architecture, it is possible to develop reusable code libraries and protocols that could be applied to any enforcement architecture that maps onto our abstract space model.

II. Concepts and Design

We begin this section by developing an intuitive understanding of proximity and realms. Once we have sketched these preliminary concepts, we define a formal proximity model and show how to map the realms to it. In doing so, we illustrate the flexibility of our model, which shows that one could adapt the same ideas to other realms of interest.

A. Intuition of Proximity

The notion of proximity can be informally defined as the nearness of two entities. These entities are active, that is they can execute actions on protected resources. Traditionally, this nearness of entities is understood in terms of physical distance, though other frames of reference, such as time, may be used. In order to use proximity as a foundational concept for access control, it is necessary to provide a formal definition that is flexible enough to accommodate various application scenarios. Before providing our definition, we will first describe five types of proximity so as to illustrate the intuition behind our formalism. Specifically, we will discuss the following types of proximity:

Geographical proximity indicates that two entities are located within a certain distance in the physical space.

Attribute-based proximity indicates that two entities share one or more common attributes, or are both located in regions of physical space that share attributes.

Social proximity indicates that two entities (represented by nodes in a social network graph) are less than a certain number of hops apart.

Cyber proximity indicates that two entities are co-present in the same online communication session.

Temporal proximity indicates that two entities are present for events separated by a limited amount of time.

1) Geographical proximity: This type of proximity is perhaps the most conventional. The entities reside at specific locations in the physical world. The distance between the entities may be measured in traditional terms, such as Euclidean distance or Manhattan distance. Alternatively, the distance may be measured in logical units that are defined based on a partitioning of the reference space; for example, in an indoor space, the number of rooms separating the two entities may quantify the distance. Regardless of the measurement used, the notion of proximity implies that the distance is less than a certain threshold value. To illustrate access control based on geographical proximity, consider a policy that specifies that users must be present in the same room. A wireless sensor network could be used to track users' positions and verify that the constraint is satisfied. Another policy could specify that users must be within a certain number of meters of each other. This policy could be enforced using a technology such as Bluetooth, which indirectly vouches for the nearness of the users.

2) Attribute-based proximity: In attribute based proximity model, each entity has a set of attributes that characterize certain properties or personal traits of this entity. These attributes can be encoded in credentials such as certificates that attest their validity. Attribute based proximity indicates the similarity of attributes of two entities. For example, a person with attribute 'Assistant Professor' is in attribute based proximity with another person with attribute 'Associate professor'. Weighting values can be associated with both the credential (*i.e.,* to specify its trustworthiness) or the trait itself (*e.g.,* to quantify the similarity between values). In an alternate view, attributes can be associated with the user's environment, such as the location in the physical world. The distance metric for proximity, then, would be an empirical measure of the difference between values, possibly weighted to reflect the veracity of credentials presented. Our work allows for both uses of attributes. Note that, unlike geographical proximity, attribute-based proximity does not imply synchronicity of actions, even if attributes of locations are used. That is, the users' presence in regions with similar attributes does not have to occur simultaneously for attribute-based proximity. To illustrate, consider an online dating service where a user can choose to share his or her profile with similar users. Potential mates with similar political views, religious backgrounds, or hometowns could be automatically granted access; such a

system would be beneficial for helping users identify potential matches more quickly.

3) Social proximity: The emergent popularity of social networks introduces a new dimension to proximity. A social network is traditionally modeled as a graph where each user is represented by a node and the connections between users are represented by edges connecting them. In the social realm, the distance metric is based on the number of hops that separate two entities within the social graph. Social proximity of two user indicates that the distance between them is less than a certain number of hops. In this case, the distance is relatively static, as changes to the distance only occur when connections between users change. Although social connections may change often, it is intuitive that the distance between any two users would change more frequently in the physical world. Policies based on social proximity are quite common. The most popular is the restriction of shared data to friends or contacts. In some cases, these restrictions can be loosened to the next step in the network, such as friends of friends. In other cases, data may be shared with other users within subnetworks; for instance, users may share data with others from the same school or employer.

4) Cyber proximity: Two users are said to be in cyber proximity if they are simultaneously involved in an online communication session. For example, users may be on the same conference call or may be chatting with one another. The distance metric could be binary, indicating co-presence in the same session, or based on degrees of separation. In the latter case, consider three users named Alice, Bob, and Charlie. If Alice and Bob are chatting while Bob is connected to a conference call with Charlie, then the distance from Alice to Charlie would be two. Alternatively, if the binary metric is used, Alice and Charlie would not be in cyber proximity, as they are not present within the same communication session.

5) Temporal proximity: While the previous notions of proximity can clearly be applied to users, temporal proximity means that two events occur within a certain relative time frame. The most natural metric would be the passage of units of time. However, in asynchronous systems, absolute time units may not be used or feasible. Instead, relative units, such as vector clocks, may be used to specify the ordering of events. In that case, the distance between two events would be the number of events that occur between them. An example of access control based on temporal proximity would be the specification of an expiration date on a contract signature. If another event, such as a signature by another party, does not occur prior to the expiration date, then the first signature is considered null and void. Another scenario where temporal proximity could be applied would be a combination of geosocial networks with missed connections.[1] When a user visits a public place, he may retrieve a token indicating his presence at that location at that time. This token could then be used to retrieve missed connections placed by others with the same token.

[1]Missed connections are popular features in publications such as alternative newspapers. One person sees another in a public place but the opportunity to meet never arises. Instead, the first person places a missed connection advertisement with enough contextual information in the hopes that the other person will read the description and desire to make contact.

B. Formal Proximity Model

Our formal definition of proximity is derived from constructing an abstract space model $\mathcal{S}$ from the *reference space models* or *realms* (*e.g.,* the physical world, social networks, communication sessions, time) identified in the previous section. Specifically, we apply the *calculus-based method* [5] that has been widely used in GIS applications. We start by showing that this approach is sufficient for modeling non-geographic reference space models.[2] We then show how it can be used for proximity-based RBAC systems.

1) Proximity model: Let $\mathcal{S}$ denote a discrete set of closed regions, called *features*, of the reference space model. For the feature $\lambda_i \in \mathcal{S}$, $\partial\lambda_i$ denotes the set of boundary points while λ° denotes the interior of the feature. Table I summarizes the formal definitions of these sets for each realm. For instance, in the geographical space, $\mathcal{S}$ would consist of regions of space that may or may not overlap; *e.g,* if λ_i is a room, then $\partial\lambda_i$ would be the points that constitute the walls.[3] The temporal realm would have events–closed time intervals–as features. Attribute-based proximity is similar, but extends the linear model to a multi-dimensional one. Features in the social realm would consist of sub-portions of the social network.

Before we elaborate on our model with additional definitions, we must address the complexity of the cyber realm. The difficulty lies in the fact that the most natural reference space model would be a hypergraph, with a hyperedge connecting all of the vertices (users) in the communication session, which cannot be directly mapped onto our abstract space model as it introduces inconsistencies in the topological relations. Our solution is to create a parallel hypergraph such that each vertex in the original is replaced by distinct vertices for each connected hyperedge. The interior would include the new vertices connected to the hyperedges of interest, and the boundary would be the other new vertices. For instance, if a user was simultaneously communicating in a Skype session and two chat sessions, then the feature λ_i containing the chat sessions would include the new vertices for the chat sessions in the interior, and the new vertex for the Skype session would be in the boundary.

Central to our model is the notion of *feature type*, which can be organized in a hierarchical manner. Table I provides examples of types for each realm. Types allow for system administrators to distinguish between, for instance, a physics exam and a chemistry exam that occur simultaneously. Feature type can be either conceptual or unit-based. Conceptual feature types assign a semantic label to a feature while unit-based

[2]While the original work only defines the method for two-dimensional geographic space, the definitions of the topological relations can be extended for multi-dimensional space, as well.

[3]Readers familiar with the calculus-based method will note that our abstract space model only focuses on area/area relationships. This is deliberate, as defining access control policies on single points or lines seems infeasible in general.

Geographical Elements of $\mathcal{S}$: Sets of points p in physical space Sample types: Room, Building, Hospital $\lambda_i = \{p \mid p$ is in a featured region $\}$, $\lambda_i^\circ = \{p \mid p$ is an interior point $\}$, $\partial\lambda_i = \{p \mid p$ is on the region's boundary $\}$
Attribute Elements of $\mathcal{S}$: Attribute vectors $\overline{a} = \langle a_1, \ldots, a_k \rangle$ representing a collection of values for considered attributes. We also write $a_i \in_A \overline{a}$ to indicate a_i is one of $a_1, \ldots, a_k$. Sample types: {Age, School}, {Age, Profession, Employer}, {Hometown} $\lambda_i = \{\overline{a} \mid \forall\ a_i \in_A \overline{a}$, a_i is within a specified range of values for that attribute$\}$ $\lambda_i^\circ = \{\overline{a} \in \lambda_i \mid \forall\ a_i \in_A \overline{a}$, a_i is strictly within ($< max$ and $> min$) the specified range$\}$ $\partial\lambda_i = \{\overline{a} \in \lambda_i \mid \exists\ a_i \in_A \overline{a}$, a_i has a borderline (maximum or minimum) value for that attribute$\}$
Social Elements of $\mathcal{S}$: Sets of edges $e \in E$ and vertices $v \in V$ such that $G = \langle V, E \rangle$ forms a social network Sample types: Friends, Colleagues, Conference attendees $\lambda_i = \{v \in V \mid v$ represents an individual in the desired group $\} \cup \{e = \langle v_1, v_2 \rangle \mid v_1 \in \lambda_i \vee v_2 \in \lambda_i\}$, $\lambda_i^\circ = \{v \in \lambda_i\} \cup \{e \in \lambda_i \mid e = \langle v_1, v_2 \rangle \wedge v_1 \in \lambda_i \wedge v_2 \in \lambda_i\}$, $\partial\lambda_i = \{e \in \lambda_i \mid e = \langle v_1, v_2 \rangle \wedge (v_1 \notin \lambda_i \vee v_2 \notin \lambda_i)\}$
Cyber Elements of $\mathcal{S}$: Sets of hyperedges $\hat{h} \in \widehat{H}$ and vertices $\hat{v} \in \widehat{V}$ given a hypergraph $G = \langle V, H \rangle$ where $h \in H$ denotes a communication session and $v \in V$ denotes a user. Definition of $\widehat{V}$: $\forall\ v_i \in V, \exists\ h_j \in H$ s.t. $v_i \in h_j \Rightarrow \hat{v}_{i,j} \in \widehat{V}$ Definition of $\widehat{H}$: $\forall\ h_i = \{v_1, \ldots, v_k\} \in H, \hat{h}_i = \{\hat{v}_{1,i}, \ldots, \hat{v}_{k,i}\} \in \widehat{H}$ Sample types: VOIP, Skype $\lambda_i = \{\hat{h}_i \mid h_i$ represents a session$\} \cup \{\hat{v}_{l,i} \in \hat{h}_i \in \lambda_i\} \cup \{\hat{v}_{l,j} \in \hat{h}_j \notin \lambda_i \mid \exists\ \hat{h}_i \in \lambda_i$ s.t. $\hat{v}_{l,i} \in \hat{h}_i\}$ $\lambda_i^\circ = \{\hat{h} \in \lambda_i\} \cup \{\hat{v}_{l,i} \in \hat{h}_i \in \lambda_i\}$ $\partial\lambda_i = \{\hat{v}_{l,j} \in \hat{h}_j \notin \lambda_i \mid \exists\ \hat{h}_i \in \lambda_i$ s.t. $\hat{v}_{l,i} \in \hat{h}_i\}$
Temporal Elements of $\mathcal{S}$: Typed time intervals $[t_i, t_j]$ Sample types: Examination, Meeting, Football game $\lambda_i = \{e \mid e$ is an event associated with some time interval $[t_i, t_j]\}$ $\lambda_i^\circ = \{t \mid t \geq t_i \wedge t \leq t_j\}$ $\partial\lambda_i = \{t_i, t_j\}$

TABLE I: Mapping of Realms to Abstract Space Model

feature types are defined by reference space, such as meters (geographical), hops (social), or minutes (temporal). Realms can have multiple units, but all units would be considered to be types, and units can only be sub-types of other units; furthermore, units would be instantiated as distinct features. For instance, in a temporal space, a feature representing 8:00:00 – 8:00:59 would denote the first minute at 8:00. Let $types$ denote the set of application-specific feature types for the realm, and let $\sqsubseteq$ denote a sub-typing partial order.

Definition 1. $\tau : \mathcal{S} \rightarrow types$ denotes a **typing function** that maps a feature in abstract space $\mathcal{S}$ to **feature type**. If $\tau(\lambda_i) = t_i$, then t_i is the **type** of λ_i.

Definition 2. $S|_t$ denotes the **restriction of features** of $S \subseteq \mathcal{S}$ to those features with a sub-type of $t_j \in t \subseteq types$:

$$S|_t = \{\lambda_i \in S \mid \exists\ t_j \in t \text{ s.t. } \tau(\lambda_i) \sqsubseteq t_j\}$$

For instance, $S|_{\{exam, mathematics\}}$ would contain only time frames representing mathematics exams in a temporal discussion. In a geographical discussion, $S|_{\{room\}}$ could denote the rooms in a building.

We can now use the notion of types, in combination with topological relations, to define our abstract distance metric. That is, let $\mathcal{T} = \{disjoint, in, touch, equal, cover, overlap\}$, the six traditional topological relations [5]. We define a *connectivity chain* as a sequence of features where no two consecutive features satisfy the $disjoint$ topological relation.

Definition 3. The sequence $\langle \lambda_0, \lambda_1, \ldots, \lambda_{n-1}, \lambda_n \rangle$ denotes a **connectivity chain** from the feature λ_0 to λ_n, such that $\neg\langle \lambda_{i-1}, disjoint, \lambda_i \rangle$ for $1 \leq i \leq n$. Let $\chi(\lambda_i, \lambda_j)$ denote the set of all connectivity chains from λ_i to λ_j, and let $\lambda_k \in c$ mean that λ_k occurs in the chain $c \in \chi(\lambda_i, \lambda_j)$.

Definition 4. $\chi|_t(\lambda_i, \lambda_j)$ denotes the **restriction of connectivity chains** connecting features λ_i and λ_j to include only intermediate features with a sub-type of $t_k \in t \subseteq types$:

$$\chi|_t(\lambda_i, \lambda_j) = \{c \in \chi(\lambda_i, \lambda_j) \mid \forall\ \lambda_k \in c, \exists\ t_l \in t \text{ s.t. } \tau(\lambda_k) \sqsubseteq t_l\}$$

Conceptual feature types provide logical measurement (where connectivity chain is a sequence of features). For instance, $\chi|_{\{room\}}(\lambda_i, \lambda_j)$ would only consist of chains of rooms that connect the two features. Alternatively, unit types provide physical measurement. For instance, $\chi|_{\{minute\}}(\lambda_i, \lambda_j)$ would contain chains whose intermediate features are the minutes that occur between the start of λ_i and the end of λ_j. Letting $\overline{c}$ denote the length of a chain (as measured in the number of intermediate features), we can define a basic distance metric as length of smallest connectivity chain connecting two features.

Definition 5. $\delta(\lambda_i, \lambda_j, t)$ denotes the **distance metric** between features λ_i and λ_j where the intermediate feature types are restricted to $t \subseteq types$ and is defined as:

$$\delta(\lambda_i, \lambda_j, t) = min(\overline{c})\ \forall c \in \chi|_t(\lambda_i, \lambda_j)$$

The final element of our proximity model is how to incorporate users. Specifically, we require some method of mapping

users to features. Let $\mathcal{U}$ denote the set of users.

Definition 6. $\mu : \mathcal{U} \rightarrow 2^{\mathcal{S}}$ denotes a **feature mapping function** that maps a user to set of features.

The power set is required for the codomain as a result of the hierarchical typing of features. For instance, a user in the social realm may belong to a group of friends, as well as a group of colleagues. Hence, $\mu(u) = \{friends, colleagues\}$. It is important to note that applying μ to the temporal realm is somewhat unintuitive. From a formal perspective, μ maps that user to *all* events in which that user participated at any time. This is due to the nature of the temporal realm. In practice, the temporal μ would restrict the focus to events within a designated time frame.

Definition 7. $\mu|_t$ denotes the **restriction of the feature mapping function** to types $t \subseteq types$ such that

$$\mu|_t(u) = \{\lambda_i \in \mu(u) \mid \exists\ t_j \in t \text{ s.t. } \tau(\lambda_i) \sqsubseteq t_j\}$$

Based on the preceding definitions, we can define a **proximity model** $\mathcal{M} = \{\mathcal{S}, \mathcal{T}, \mathcal{U}, \tau, \mu, \delta\}$.

2) Role Proximity: Using the model $\mathcal{M}$, we can define the notion of **role proximity**. We start with the traditional RBAC concepts of roles ($\mathcal{R}$) and users ($\mathcal{U}$). When a user logs into the system, he is associated with a new session. Let SES denote the set of sessions, $SU : SES \rightarrow \mathcal{U}$ the mapping of sessions to users, $SR : SES \rightarrow 2^{\mathcal{R}}$ the mapping of sessions to *possible* roles that could be activated, and $Act : \mathcal{U} \rightarrow 2^{\mathcal{R}}$ the mapping of users to active roles. Observe that, for any $u \in \mathcal{U}$

$$Act(u) \subseteq \bigcup_{s \in SU^{-1}(u)} SR(s)$$

where $SU^{-1}(u)$ denotes the preimage of u under SU, *i.e.,* the set of sessions associated with the user. That is, every one of a user's active roles must be associated with some session. We can define two distinct types of role proximity using these definitions.

Definition 8. A user $u \in \mathcal{U}$ is said to be in (t_1, d, t_2)-**weak role proximity** ((t_1, d, t_2)-wrp) of a role r for $t_1, t_2 \in types$ and $d \in \mathbb{R}^+$ if $\exists \hat{u} \in \mathcal{U}, \hat{u} \neq u$ such that all of these hold:

1) $r \in Act(\hat{u})$
2) $\lambda_i \in \mu|_{\{t_1\}}(u)$
3) $\lambda_j \in \mu|_{\{t_1\}}(\hat{u})$
4) $\delta(\lambda_i, \lambda_j, t_2) \leq d$

Weak role proximity, then, considers only users' active roles. Observe that two feature types are necessary, as the unit separating the features will most likely have a different type than the features themselves. For instance, in social proximity, a manager at one company may be in $(org, 1, friend)$-wrp of the CTO of another company if there are employees of both companies that are friends. In a temporal setting, if a user signs a document at some meeting, $(meeting, 4, hour)$-wrp is satisfied if a manager signs the document at another meeting with no more than 4 hours separating the meetings.

Definition 9. A user $u \in \mathcal{U}$ is said to be in (t_1, d, t_2)-**strong role proximity** ((t_1, d, t_2)-srp) of a role r for $t_1, t_2 \in types$ and $d \in \mathbb{R}^+$ if $\exists \hat{u} \in \mathcal{U}, \hat{u} \neq u$ such that all of these hold:

1) $r \in \bigcup_{s \in SU^{-1}(\hat{u})} SR(s)$
2) $\lambda_i \in \mu|_{\{t_1\}}(u)$
3) $\lambda_j \in \mu|_{\{t_1\}}(\hat{u})$
4) $\delta(\lambda_i, \lambda_j, t_2) \leq d$

That is, strong role proximity considers roles that *could* be activated during some session for the user, but may not currently be. The rationale for strong role proximity is that it may be desirable to base policies on roles that are not currently active. For instance, if a military environment demands that there are no civilians present, strong role proximity can be used to meet this demand, because it does not require users to explicitly activate the civilian role.

3) Proximity Constraints: Using $\mathcal{M} = \{\mathcal{S}, \mathcal{T}, \mathcal{U}, \tau, \mu, \delta\}$ and the definitions above, we can now define **proximity constraints** that can be used in a policy language. Our language is similar to that defined in [17], except that we remove the assumption of geographical proximity and spatial roles. The simplified grammar for a proximity constraint clause is written as:

$$\begin{array}{lll} C & ::- & C \vee C \\ & — & C \wedge C \\ & — & \neg C \\ & — & S \quad Q \quad n \quad role \quad unit \quad thr \\ S & ::- & weak \mid strong \\ Q & ::- & at\ most \mid at\ least \mid \epsilon \end{array}$$

The semantics of such a constraint dictate that satisfaction requires separate users. That is, the semantics for the basic constraint $(weak\ n\ r\ unit\ thr)$ dictate that there is a set $\widehat{U} \subseteq \mathcal{U}$ such that

1) $|\widehat{U}| = n$
2) $(t, thr, unit)$-wrp holds for some type $t \in types$
3) $\forall u \in \widehat{U}\ r \in Act(u)$
4) $\forall u \notin \widehat{U}\ r \notin Act(u)$

Semantics for the strong variant would replace the last two criteria as

3) $\forall u \in \widehat{U}\ \exists s \in SU^{-1}(u)$ such that $r \in SR(s)$
4) $\forall u \notin \widehat{U}\ \nexists s \in SU^{-1}(u)$ such that $r \in SR(s)$

Semantics for the other possible constraints are straightforward variations. Note that t is specified independently of the proximity constraint and is determined according to the remainder of the policy. Let $\mathcal{C}$ denote the set of proximity constraints in this language.[4]

4) Proximity-based RBAC model: We can now conclude this section with our formal definition of a proximity-based RBAC model. Let $\mathcal{M} = \{\mathcal{S}, \mathcal{T}, \mathcal{U}, \tau, \mu, \delta\}$ denote a proximity model as defined previously. Policies would be based on **proximity tuples** $p = \langle r, t, c \rangle$, where $c \in \mathcal{C}$ is a proximity constraint, $t \in types$ is a type associated with the requesting

[4]Observe that this syntax only supports a single realm per constraint. Intuitively, the syntax could be extended to specify the realm and the type t within the constraint. This would allow for complex policies that consider multiple dimensions (*e.g.,* a policy could simultaneously have spatial, temporal, and social constraints). As each realm would define its own distance metric δ, we believe this approach is feasible. However, we have not fully considered the implications of this approach, and leave such composition of proximity realms for future work.

Geographical

Example: An officer is allowed to read a secret file only if no civilian is present within 500m and at least one senior officer is present in the same room.
$types = \{room, meters\}$, $\mathcal{O} = \{SecretFile\}$, $\mathcal{A} = \{read\}$, $\mathcal{R} = \{SeniorOfficer, Officer, Civilian\}$
Proximity Constraints $C_1 = \langle strong, at\ most, 0, Civilian, meters, 500\rangle$,$C_2 = \langle weak, at\ least, 1, SeniorOfficer, room, 0\rangle$
Proximity tuple $p = \langle Officer, room, C_1 \wedge C_2\rangle$
Policy: $\{p, read, SecretFile\}$

Attribute

Example: A dating site member can view my profile if they have same profession and are no more than 10 years older.
$types = 2^{\{profession, age\}}$, $\mathcal{O} = \{MyProfile\}$, $\mathcal{A} = \{view\}$, $\mathcal{R} = \{Member, Self\}$
Proximity Constraints $C_1 = \langle weak, \epsilon, 1, Self, \{profession\}, 0\rangle$, $C_2 = \langle weak, \epsilon, 1, Self, \{age\}, 10\rangle$
Proximity tuple $p = \langle Member, \{profession, age\}, C_1 \wedge C_2\rangle$
Policy: $\{p, view, MyProfile\}$

Social

Example: A member of IEEE network is allowed to view my conference album only if he is a friend of a friend or closer.
$types = \{individual, network, hops\}$, $\mathcal{O} = \{ConfAlbum\}$, $\mathcal{A} = \{view\}$, $\mathcal{R} = \{Self, IEEEMember\}$
Proximity Constraints $C = \langle strong, \epsilon, 1, Self, hops, 2\rangle$
Proximity tuple $p = \langle IEEEMember, individual, C\rangle$
Policy: $\{p, view, ConfAlbum\}$

Cyber

Example: A manager can edit a shared Google document only if he is in a GoogleTalk session with a senior manager.
$types = \{GoogleTalk\}$, $\mathcal{O} = \{document_1\}$, $\mathcal{A} = \{write\}$, $\mathcal{R} = \{Manager, SeniorManager\}$
Proximity Constraints $C = \langle weak, at\ least, 1, SeniorManager, GoogleTalk, 0\rangle$
Proximity tuple $p = \langle Manager, GoogleTalk, C\rangle$
Policy: $\{p, write, document_1\}$

Temporal

Example: A supervisor can only sign an employee's time card within 24 hours after the employee did.
$types = \{hours, card\ signature\}$, $\mathcal{O} = \{time\ card\}$, $\mathcal{A} = \{sign\}$, $\mathcal{R} = \{Employee, Supervisor\}$
Proximity Constraints $C = \langle weak, at\ least, 1, Employee, hours, 24\rangle$
Proximity tuple $p = \langle Supervisor, card\ signature, C\rangle$
Policy: $\{p, sign, time\ card\}$

TABLE II: Example policies for various realms

user's feature, and r is the requested role. Specifically, if $\mathcal{P}$ denotes the set of all such tuples, $\mathcal{A}$ denotes the set of actions, and $\mathcal{O}$ denotes the set of objects, a **proximity-based RBAC policy** would be the relation $Pol : \mathcal{P} \times \mathcal{A} \times \mathcal{O}$. That is, a policy specifies the actions allowed on an object, such that the proximity constraint (which includes the subject's role) is satisfied. The **proximity-based RBAC model** Φ would consist of the set of all such policies. Table II presents examples of policies for the five proximity realms.

III. Enforcement

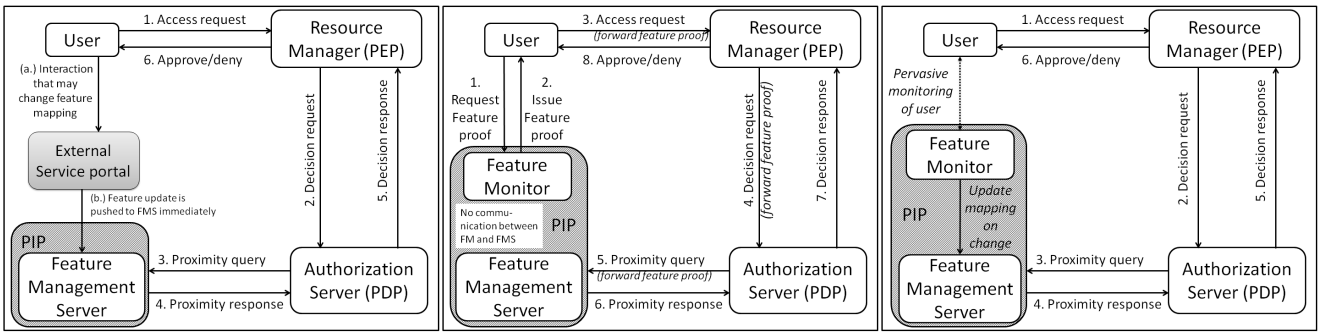
Designing a generic architecture that works across different applications and realms is a crucial but challenging task. Different types of proximity and organizational settings have different requirements and a single architecture may not work for all cases. However, if an architecture is defined carefully then a major part of it can be common and only a small portion of it may need to be changed across realms. For instance, the method for acquiring feature mapping for a user is realm-specific. We propose a generic architecture and discuss changes that are required in it to accommodate different types of feature acquisition and communication strategies. Further, we prove some properties of this system and discuss its limitations.

A. System architecture

In this section, we will define an enforcement architecture for enforcing proximity-based RBAC. While different application scenarios will employ different technologies, our goal in this section is to highlight common features of principals and define required behaviors. The purpose in defining such an abstract architecture is to establish a framework for reasoning about the feasibility of designing and building proximity-based RBAC systems.

For simplicity, we assume a centralized server with universal knowledge of the user-feature mapping. In our current approach, we emphasize the necessity of correctly mapping each user to a feature (or a set of features) in the reference space model. We refer to this process as *feature attestation*, and is the responsibility of the Policy Information Point (PIP) [18]. Feature attestation could be accomplished using cryptographic techniques, such as digitally signed proofs of location, timestamps, or credentials. The Policy Decision Point (PDP) uses the result of the constraint evaluation to facilitate the proper functioning of the Policy Enforcement Point (PEP).

- *User*: The *User* represents the entity that is assigned roles and initiates access request.
- *Feature management server (FMS)*: This server maintains the current feature mapping of every user in the system. Given a *proximity query*, in which the PDP submits



(a) No FM, External portal (b) Independent FM, No communication (c) FM direct communication

Fig. 1: Enforcement architecture

a proximity constraint and the requesting user, FMS computes the proximity distances and determines if the constraint is satisfied. It serves as the main component of the PIP and responds to queries from the authorization server.

- *Feature monitor (FM)*: This optional component is used to communicate with the user as a means of maintaining the feature mapping. If present, this component may issue a *feature proof* to the user, such as a digital certificate avowing the claimed feature, that the user can submit as a credential along with the request as shown in Figure 1b. In an alternate architecture (Figure 1c), FM may pervasively monitor the user and communicate with the FMS to ensure the user-feature mapping is updated in a timely manner. Alternatively, this monitor may be absent entirely, in which case the user would communicate with an external portal that pushes feature update to FMS as discussed later (Figure 1a).
- *Authorization server (AS)*: This serves as PDP and is responsible for evaluating policy. It consults FMS by issuing proximity queries. Using the results of the queries, it evaluates the remainder of the policy and determines if the request is to be granted.
- *Resource manager (RM)*: The resource manager serves as the PEP and is responsible for controlling access to protected resources. The resource manager may hold the resources itself, or it may serve as a ticket-granting service.

In some cases, an external *service portal*, which is a trusted third party, can replace the FM. For instance, in social proximity, the proximity-based RBAC system may rely on an independent social network service. That is, the proximity-based RBAC system consults the external social network and overlays the feature mapping on top of the existing network. In these types of cases, the user interacts with the service portal to make changes, and the service portal pushes these updates to the proximity-based RBAC system. This is the architecture shown in Figure 1a.

1) Feature acquisition and communication: Most of the interaction between principals is straightforward and functions like a typical RBAC architecture that consists of users, PEP, PDP, and PIP. What is unique about proximity-based RBAC is the acquisition and communication of feature mapping that is achieved via the interaction between users and the PIP. Although the precise interaction would be application specific, we identify three fundamental approaches that are illustrated in Figure 1.

1) **No FM, external portal** – In this approach, illustrated in Figure 1a, the user explicitly interacts with an external service portal (*e.g.,* a social network web site or a trusted third-party attribute certification service) that is independent of the proximity-based RBAC system in order to update his or her associated feature(s). For instance, in social proximity, the user makes changes to his or her profile in a social network application, and these changes are pushed to the FMS by the application. In temporal proximity, events are logged by some application, and the FMS receives this data accordingly. This approach is applicable for all realms, though the geographical realm is challenging, as users typically do not have to interact with a centralized software portal in order to move. Instead, the other two approaches more accurately describe approaches for geographical proximity.
2) **Independent FM, no communication** – In this approach, users interact with a distributed set of devices (feature monitors) that have no direct communication links to the FMS. Instead, the devices provide the user with a credential (feature proof) that asserts the correct feature mapping. User includes this feature proof in access request and can be validated by the FMS as shown in Figure 1b. For instance, in geographical proximity, the user may have a Bluetooth-enabled device that exchanges data with a receiver as the user moves. As the user moves, the credential updates are performed locally, and only pushed to the FMS when the user makes a request. As such, in order to enforce proximity constraints correctly, the system must force users to push their credentials sufficiently often. For instance, in the geographic realm, doors separating rooms may be considered objects. Thus, in order for the user to change

features (*i.e.,* move from one room to another), he must push his credentials before the door can be unlocked.

3) **FM direct communication** – In this approach, a distributed sensor network (FM) continually monitors changes to the user's feature mapping. When the mapping changes, the sensor pushes the updated information to the FMS accordingly (refer Figure 1c). This approach is more appropriate for real-time geographical proximity This approach is also good for temporal proximity. For instance, the sensor may consist of a program that monitors updates to log files, and sends updates when the file changes.

In all above communication protocols, FMS is responsible for evaluating if the proximity constraints are satisfied. In the subsequent section, we present an algorithm for evaluating these constraints and discuss its complexity.

B. Complexity analysis

Algorithm 1 describes one approach to evaluating a single complex proximity constraint. Specifically, assume the constraint $c \in \mathcal{C}$ consists of m primitive weak proximity constraints, each of the form $(weak\ n\ role\ unit\ thr)$.[5] This algorithm will evaluate each primitive constraint to yield a Boolean value to replace the constraint. Once all constraints are evaluated, the resulting Boolean expression is evaluated. If the return value is **true**, then the constraint was satisfied. Note that handling variations allowed by the policy language involves trivial changes that do not affect the complexity of the algorithm.[6]

Let $\mathcal{D}$ denote the decision problem that answers whether or not a proximity constraint can be satisfied. That is, assume μ maps a user to a feature in polynomial time. Then $\mathcal{D}$ takes as input $\mathcal{M} - \mu$ (*i.e.,* the model with no current mapping of users to features) and a policy $p \in \Phi$. $\mathcal{D}$ returns "yes" if there exists a mapping μ such that the proximity constraint $c \in \mathcal{C}$ in the tuple $\langle r, t, c\rangle$ for the policy p is true. If no such mapping exists, $\mathcal{D}$ returns "no."

Lemma 1: Given a candidate mapping μ and a distance function δ that run in polynomial time, verifying μ satisfies p can be done in polynomial time. That is, $\mathcal{D}$ is in NP.

Proof: Let n be the maximum of $|\mathcal{U}|$, $|\mathcal{R}|$, and m, where m is the number of primitive constraint clauses in p. Executing Algorithm 1 without the mapping $\mu|_t(t)$ can be done in $\mathcal{O}(n^3)$ time.

Theorem 1: $\mathcal{D}$ is NP-hard.

Proof: Our proof is by reduction from Boolean satisfiability (SAT). Given an arbitrary Boolean expression, one can replace each independent variable with a unique primitive proximity constraint in polynomial time. Based on the complexity of SAT [6], $\mathcal{D}$ is NP-hard. □

[5]For simplicity, we ignore the use of parentheses to shape the Boolean expression.

[6]For instance, supporting *at most* and *at least* requires adding **else** checks to the final **if-then-else** block. These cases can be enumerated and do not vary with the size of n.

Algorithm 1: Evaluate $(weak\ n\ role\ unit\ thr)$ constraints

Input: $c \in \mathcal{C}$: a proximity constraint, consisting of m primitive constraints, joined using Boolean connectives ; $u \in \mathcal{U}$: the requesting user ; $t \in \mathcal{T}$: requesting user's feature type

Output: **true** or **false**

```
/* break c into its primitive
constraints */
⟨c_1, ..., c_m⟩ ← c
Feature_u ← μ|_{t}(u)
ActiveRoles ← ∅
for c_i := c_1 to c_m do
    Matches ← 0
    foreach û ∈ U − {u} do
        /* weak proximity semantics */
        foreach r ∈ Act(û) do
            if r = c_i.role then
                Feature_o ← μ|_t(û)
                distance ←
                δ(Feature_u, Feature_o, c_i.unit)
                if distance ≤ c_i.thr then
                    Matches ← Matches + 1
    if Matches = c_i.n then
        b_i ← true
    else
        b_i ← false
return EvaluateBoolean ⟨b_1, ..., b_m⟩
```

Corollary 1.1: Given a candidate mapping μ and a distance function δ that run in polynomial time, $\mathcal{D}$ is NP-complete.

Proof: From Theorem 1, $\mathcal{D}$ is NP-hard. Under the assumption of polynomial run time for μ and δ, by Lemma 1, $\mathcal{D}$ is in NP. Thus, it is NP-complete. □

These complexity results illustrate a warning for building and maintaining proximity-based RBAC systems. Clearly, the latter result shows that attempting to build an automated tool that determines if a set of policies can be evaluated would require heuristics to be tractable. Furthermore, the complexity of Algorithm 1, while polynomial-time, is not particularly efficient and may present scaling challenges. Thus, designers of proximity-based systems should plan carefully to streamline the operation of the PIP.

C. Properties of protocols

Before describing a general approach to constructing enforcement protocols, we first present some theoretical results that illustrate the limitations of such systems. In real systems, communication between various components of the architecture may entail some delay. This communication delay may

lead system into a state where the evaluation of proximity constraints at a certain time is not consistent with the current feature mapping of users. For example, the feature mapping of a user involved in a proximity constraint may change while the constraint is still being evaluated by FMS. The following results use impossibility of distributed consensus [10] to show that correct evaluation of constraints cannot be guaranteed unless FMS has correct mapping of all users and these mapping don't change until FMS has completed the evaluation of proximity constraint. Theorem 2 presents the proof for the deployment scenario illustrated in Figure 1a. Corollary 2.1 presents this proof for the scenario in Figure 1b, while the Figure 1c case is handled by Theorem 3.

Theorem 2: Given a deployment with no feature monitor such that μ is updated only through explicit interaction with a service portal. Correct proximity constraint evaluation can be enforced only if access to the service portal (by the users and FMS) is synchronous.

Proof: Assume that evaluation can be enforced correctly. To prove that access must be synchronous, we will map proximity constraint evaluation onto a consensus protocol P. Specifically, let $p_1, \ldots, p_n$ denote asynchronous processes representing users and the service portal would consist of a buffer for P. Each p_i for a user would respond with a 1 if the user's feature has changed, 0 if unchanged, and b denotes the request is still pending. The goal of P would be to have a response of 0 for all users, indicating that the portal has the correct mapping of users to features. However, if a single p_i fails without notice (*e.g.,* the user's network connection gets dropped), then no such P can exist [10]. Thus, if users are granted asynchronous access to the service portal, no protocol involving the portal and FMS can exist that guarantees constraint evaluation is correct. Therefore, by contradiction, correct evaluation requires synchronous access. □

Corollary 2.1: Given a deployment with asynchronous feature monitors, correct proximity constraint evaluation cannot be enforced.

Proof: Similar to the preceding.

The above results address the effect of limitations of communication channel between users and FM/FMS on correct evaluation of proximity constraints. Inconsistency in constraint evaluation may also stem from asynchronicity of communication between the components of our architecture. That is, assuming that the channel between users and FM/FMS is synchronous and FMS has all correct mappings, it is still not possible to achieve correct evaluation of proximity constraints. This is because by the time the proximity evaluation decision reached RM and RM accepts/denies the request, the feature mapping of some user may have changed in a way that it changes the outcome of proximity constraint evaluation. The following theorem proves this result.

Theorem 3: Assuming communication between RM, AS, and FMS is asynchronous, it is impossible for a deployment with feature monitors to guarantee correct evaluation of proximity constraints, even if the monitors have synchronous access to FMS.[7]

Proof: Similar to the preceding Lemma, except the consensus protocol is now to be executed between the principals of our architecture. That is, as communication between RM, AS, and FMS is asynchronous, these three principals cannot achieve consensus. Formally, let $\sigma = \langle \sigma_1, \ldots, \sigma_n \rangle$ denote a sequence of events in the evaluation of the constraint and the resulting data exchange. Assume FMS completes evaluation at σ_i and sends the result at σ_{i+1} to AS, who forwards the result to RM at σ_{i+2}. Let $\widehat{\sigma_{i+2}}$ denote the reception by FMS of a message from some FM that would change the result of the proximity evaluation. As communication between FMS and FM is independent from communication between AS and RM, $\widehat{\sigma_{i+2}}$ can occur simultaneously as σ_{i+2}. As such, when RM grants (or denies) access at σ_{i+3}, the proximity constraint may evaluate to a different value. Hence, the principals cannot achieve consensus, and correct policy enforcement cannot be guaranteed. □

We wish to emphasize that these impossibility results do not mean that one cannot build a proximity-based RBAC system that functions correctly. Rather, any such system will have brief moments when policy decisions will be incorrect. Specifically, when a user-feature mapping changes at the same time that a related constraint is evaluated, a race condition occurs. For instance, in geographical proximity, if a policy that requires the presence of a supervisor is evaluated immediately after the supervisor enters the room, it is possible that the system would have a false negative, denying access unnecessarily, as the supervisor's new location had not been propagated to the FMS yet. Thus, designers of proximity-based RBAC systems should account for such cases.

Best-guess protocols: Despite these impossibility results, system designers can achieve generally accurate proximity constraint evaluation, provided one can tolerate brief moments of erroneous results. We refer to this phenomenon as a *best-guess assumption*. The general approach is that communication proceeds as illustrated in Figure 1. In addition, FMS stores a cache of the most recent proximity queries for continual re-evaluation over a designated period of time. The frequency of the re-evaluation would be an application-specific parameter. Within the designated time window, if the constraint evaluation changes, FMS would forward this new information to PDP. If this result changes the policy decision, PDP would inform the PEP, which would revoke access accordingly.

IV. Related work

Role based access control (RBAC) [20] is a permission model that grants access based on roles that users have as a part of an organization. Several extensions to RBAC have

[7] One should be careful to note that Theorems 2 and 3 are not contradictory. Rather, Theorem 2 disproves, in essence, the *converse* of Theorem 3.

been proposed that attempt to incorporate various contextual factors while making access decisions. Previous works have incorporated location [8], [16], [13], [1], [19], [3], [2], [7], [4] and time [1], [3] of the user requesting access as a factor in decision making. Prox-RBAC [17] extended the notion of spatially aware RBAC to consider the relative locations of other users within an indoor space model [15], [14], and is the closest paper to the current work. However, Prox-RBAC relied on an intuitive, informal notion of proximity that allowed for surprising and contradictory interpretations of proximity; furthermore, Prox-RBAC focused exclusively on the geographic realm, whereas our own work is applicable to a wider range of contextual factors.

While Prox-RBAC is unique in combining proximity constraints with RBAC, it is not the first work to consider contextual similarity between users when requests are evaluated. TMAC [11] incorporates contextual information into team-based access control by actively monitoring ongoing interactions. PBAC [9], [12] models focus on efficiently granting authorizing emergency service providers in time-critical settings. However, all of these works are restricted to the geographic realm, unlike our own.

V. Conclusion

In this paper we have explored various notions of proximity. Specifically, we have discussed five types of proximity: geographical, attribute-based, cyber, social and temporal. We have developed a formal model of proximity that is generic enough to specify all these types of proximity. We have presented theoretical results illustrating the challenges inherent in implementing a proximity-based RBAC system, and we have also described approaches to overcome these difficulties. In summary, we argue that it is feasible to deploy a practical proximity-based RBAC system for a variety of contextual factors.

References

[1] S. Aich, S. Sural, and A. K. Majumdar. STARBAC: Spatiotemporal role based access control. In *OTM Conferences*, 2007.

[2] V. Atluri and S. A. Chun. A geotemporal role-based authorisation system. In *International Journal of Information and Computer Security*, volume 1, pages 143–168, 2007.

[3] S. Chandran and J. Joshi. LoT RBAC: A location and time-based RBAC model. In *Proc. of 6th International Conference on Web Information Systems Engineering (WISE)*, pages 361–375. Springer-Verlag, 2005.

[4] L. Cirio, I. F. Cruz, and R. Tamassia. A role and attribute based access control system using semantic web technologies. In *Proc. of 2007 On the Move to Meaningful Internet Systems - Volume Part II*, OTM'07, pages 1256–1266, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] E. Clementini, P. D. Felice, and P. v. Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *Proc. of 3rd International Symposium on Advances in Spatial Databases (SSD)*, pages 277–295, London, UK, 1993. Springer-Verlag.

[6] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. of 3rd Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[7] I. F. Cruz, R. Gjomemo, B. Lin, and M. Orsini. A location aware role and attribute based access control system. In *Proc. of 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*, pages 84:1–84:2, New York, NY, USA, 2008. ACM.

[8] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca. GEO-RBAC: A spatially aware RBAC. In *ACM Transactions on Information and System Security (TISSEC)*, 2007.

[9] S. M. Didar-Al-Alam, H. Mahmud, and M. A. Mottalib. Modifications in proximity based access control for multiple user support. *International Journal of Engineering Science and Technology*, 2:3603–3613, 2010.

[10] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32:374–382, April 1985.

[11] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible team-based access control using contexts. In *Proc. of 6th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 21–27, New York, NY, USA, 2001. ACM.

[12] S. K. S. Gupta, T. Mukherjee, K. Venkatasubramanian, and T. B. Taylor. Proximity based access control in smart-emergency departments. In *Proc. of 4th Annual IEEE international Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, pages 512–, Washington, DC, USA, 2006. IEEE Computer Society.

[13] F. Hansen and V. Oleschuk. SRBAC: A spatial role-based access control model for mobile systems. In *Proc. of 8th Nordic Workshop on Secure IT Systems (NORDSEC)*, pages 129–141, October 2003.

[14] C. S. Jensen, H. Lu, and B. Yang. Graph model based indoor tracking. In *10th International Conference on Mobile Data Management (MDM)*, pages 122–131, 2009.

[15] C. S. Jensen, H. Lu, and B. Yang. Indoor–a new data management frontier. *IEEE Data Eng. Bull.*, 33(2):12–17, June 2010.

[16] M. S. Kirkpatrick and E. Bertino. Enforcing spatial constraints for mobile rbac systems. In *Proc. of 15th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 99–108, New York, NY, USA, 2010. ACM.

[17] M. S. Kirkpatrick, M. L. Damiani, and E. Bertino. Prox-RBAC: A proximity-based spatially aware RBAC. In *Proc. of 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, pages 339–348, 2011.

[18] Oasis. Oasis extensible access control markup language (xacml) tc. *Spring*, 2009(May 5):1–16, 2004.

[19] I. Ray, M. Kumar, and L. Yu. LRBAC: A location-aware role-based access control model. In *Proc. of Internation Conference on Information Systems Security (ICISS)*, pages 147–161, 2006.

[20] R. Sandhu. Role-based access control models. In *IEEE Computer*, Feb. 1996.