Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

A formal proximity model for RBAC systems

Aditi Gupta [a,*], Michael S. Kirkpatrick [b], Elisa Bertino [a]

[a] Department of Computer Science, Purdue University, West Lafayette, IN 47907, United States
[b] Department of Computer Science, James Madison University, Harrisonburg, VA 22807, United States

ARTICLE INFO

Article history:
Received 22 May 2013
Received in revised form
20 August 2013
Accepted 31 August 2013

Keywords:
Access control
Security
Mobility
Context awareness
Proximity

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, predefined 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.

1. 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 (Damiani et al., 2007; Aich et al., 2007; Atluri and Chun, 2007).

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 in the same room as the employee may be more important than just ensuring that the employee is present in the store office. To enable the creation of such policies, Prox-RBAC (Kirkpatrick et al., 2011) 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.

* *Corresponding author*. Tel.: +1 7654969398.
E-mail addresses: aditi@purdue.edu (A. Gupta), kirkpams@jmu.edu (M.S. Kirkpatrick), bertino@purdue.edu (E. Bertino).
0167-4048/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2013.08.012

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 h. 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 (Clementini et al., 1993) 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 propose three enforcement architectures in this work to accommodate different types of feature acquisition and communication approaches. We also provide templates for enforcement protocols for these architectures, formalize these protocols using PCL (Datta et al., 2007) and prove security properties of these protocols. In doing so, we also highlight the theoretical limits of correct enforcement of proximity constraints.

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.

2. 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.

2.1. *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.

2.1.1. *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.

2.1.2. *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). As another example, 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. In an alternate view, attributes can be associated with the user's environment, such as the type of 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. For instance, if two users are in restaurants, possible attributes may be the type of restaurant or the name of the chain; if the restaurants share the same parent company, they would be considered to be in close proximity, regardless of their physical distance. Other attributes could be the presence of public wi-fi, the temperature of the surrounding area, or the most popular professional sports team of the area. Our work allows for both uses of attributes, either relative to the user or the context.

2.1.3. *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 sub-networks; for instance, users may share data with others from the same school or employer.

2.1.4. *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.

2.1.5. *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.

2.2. *Hybrid proximity realms*

Although we do not explicitly model the case, we posit that it would be possible to create policies for hybrid realms that combine two or more of the above mentioned realms. For instance, one could consider a realm that combines military ranks, the bases to which the officers are assigned, and their connections within particular social networks. Such a multidimensional policy model would combine elements of attribute, geographical and social proximities. While our model is sufficient to define such a hybrid realm (*i.e.*, by using appropriate topological relations), crafting appropriate distance metrics – by mapping realms to a multidimensional coordinate system – would be application specific. We find attempts at formalizing such a meta-model to be needlessly complex, and omit this case from further consideration for the present work.

2.3. *Formal proximity model*

Our formal definition of proximity is derived from constructing an abstract space model $\mathscr{S}$ 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* (Clementini et al., 1993) that has been widely used in GIS applications. We start by showing that this approach is sufficient for modeling non-geographic

[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.

reference space models.[2] We then show how it can be used for proximity-based RBAC systems.

2.3.1. Proximity model

Let $\mathscr{S}$ denote a discrete set of closed regions, called *features*, of the reference space model. For the feature $\lambda_i \in \mathscr{S}$, $\partial\lambda_i$ denotes the set of boundary points while λ° denotes the interior of the feature. Table 1 summarizes the formal definitions of these sets for each realm. For instance, in the geographical space, $\mathscr{S}$ 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 1 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 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. $\tau : \mathscr{S} \rightarrow types$ denotes a **typing function** that maps a feature in abstract space $\mathscr{S}$ to **feature type**. If $\tau(\lambda_i) = t_i$, then t_i is the **type** of λ_i.

The abstract space model can be restricted to only contain features that have certain types. This may be useful for applications that are only interested in certain types of features but not others.

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

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

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. We use a set of six topological relations defined in Clementini et al. (1993) to specify the relationships between features of the abstract space. Let $\mathscr{T}$ be this set of topological relations and is defined as

$$\mathscr{T} = \{disjoint, in, touch, equal, cover, overlap\}$$

We define a *connectivity chain* as a sequence of features where no two consecutive features satisfy the *disjoint* topological relation.

Definition 3. The sequence $<\lambda_0, \lambda_1, \ldots, \lambda_{n-1}, \lambda_n>$ 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 $t_k \in t \subseteq types$:

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

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 $\mathscr{U}$ denote the set of users.

[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.

Table 1 – Mapping of realms to abstract space model.

Geographical
Elements of $\mathscr{S}$: Sets of points p in physical space
Sample types: Room, Building, Hospital
$\lambda_i = \{p | p$ is in a featured region$\}$,
$\lambda_i^\circ = \{p | p$ is an interior point$\}$,
$\partial\lambda_i = \{p | p$ is on the region's boundary$\}$

Attribute
Elements of $\mathscr{S}$: 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}
$\lambda_i = \{\overline{a} | \forall a_i \in_A \overline{a}$, a_i is within a specified range for that attribute$\}$
$\lambda_i^\circ = \{\overline{a} \in \lambda_i | \forall a_i \in_A \overline{a}$, a_i is strictly within the specified range$\}$
$\partial\lambda_i = \{\overline{a} \in \lambda_i | \exists a_i \in_A \overline{a}$, a_i has a borderline (maximum or minimum) value for that attribute$\}$

Social
Elements of $\mathscr{S}$: 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
$\lambda_i \{v \in V | v$ is an individual$\} \cup \{e = \langle v1, v2 \rangle | v_1 \in \lambda_i \vee v_2 \in \lambda_i\}$,
$\lambda_i^\circ = \{v \in \lambda_i\} \cup \{e \in \lambda_i | e = \langle v_1, v_2 \rangle \wedge v_1 \in \lambda_i \wedge v_2 \in \lambda_i\}$,
$\partial\lambda_i = \{e \in \lambda_i | e = \langle v_1, v_2 \rangle \wedge (v_1 \notin \lambda_i \vee v_2 \notin \lambda_i)\}$

Cyber
Elements of $\mathscr{S}$: Sets of hyperedges $\widehat{h} \in \widehat{H}$ and vertices $\widehat{v} \in \widehat{V}$ given a hypergraph $G = \langle V, H \rangle$ where $h \in H$ denotes a communication session and $v \in V$ denotes a user, where
$\widehat{V} \triangleq \{\widehat{v}_{i,j} \in \widehat{V} | \exists v_i \in V, h_j \in H$ s.t. $v_i \in h_j\}$
$\widehat{H} \triangleq \{\widehat{h}_i = \{\widehat{v}_{1,i}, \ldots, \widehat{v}_{k,i}\} \in \widehat{H} | \exists h_i = \{v_1, \ldots, v_k\} \in H\}$
Sample types: VOIP, Skype
$\lambda_i = \{\widehat{h}_i | h_i$ represents a session$\} \cup \{\widehat{v}_{l,i} \in \widehat{h}_i \in \lambda_i\} \cup \{\widehat{v}_{l,j} \in \widehat{h}_j \notin \lambda_i | \exists\ \widehat{h}_i \in \lambda_i$ s.t. $\widehat{v}_{l,i} \in \widehat{h}_i\}$
$\lambda_i^\circ = \{\widehat{h} \in \lambda_i\} \cup \{\widehat{v}_{l,i} \in \widehat{h}_i \in \lambda_i\}$
$\partial\lambda_i = \{\widehat{v}_{l,j} \in \widehat{h}_j \notin \lambda_i | \exists \widehat{h}_i \in \lambda_i$ s.t. $\widehat{v}_{l,i} \in \widehat{h}_i\}$

Temporal
Elements of $\mathscr{S}$: Typed time intervals $[t_i, t_j]$
Sample types: Examination, Meeting, Football game
$\lambda_i = \{e | e$ is an event associated with some time interval $[t_i, t_j]\}$
$\lambda_i^\circ = \{t | t \geq t_i \wedge t \leq t_j\}$
$\partial\lambda_i = \{t_i, t_j\}$

Definition 6. $\mu : \mathscr{U} \to 2^{\mathscr{S}}$ 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

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

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

2.3.2. *Role proximity*

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

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

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 $u \in \mathscr{U}$ 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 $\exists \widehat{u} \in \mathscr{U}, \widehat{u} \neq u$ such that all of these hold:

1. $r \in Act(\widehat{u})$
2. $\lambda_i \in \mu|_{\{t1\}}(u)$
3. $\lambda_j \in \mu|_{\{t_1\}}(\widehat{u})$
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 h separating the meetings.

At this point, it is necessary to point out that the temporal realm presents a unique complication for our definitions as written. Specifically, it is possible that r is no longer in $Act(\hat{u})$ at the time that the constraint needs to be evaluated for user u, although the proximity constraint should be considered satisfied. The solution, then, is to emphasize that $Act(\hat{u})$ is evaluated at the time that $\hat{u}$ performs some action.[4] For instance, in the preceding example, both the user and the manager must perform the action of signing the document. This modeling choice is, in essence, syntactic sugar that allows us to use consistent terminology.

This interpretation presents a clear engineering challenge, which is determining how much information about session mappings must be maintained over time. If all temporal proximity constraints require users to perform actions, then the system must only log events that occur. On the other hand, if the constraints are passive, *i.e.*, at least one of the user is not required to perform an explicit action, then the administrative overhead would be higher – possibly prohibitively high. Consequently, system designers would have to make appropriate choices for their specific applications.

Definition 9. A user $u \in \mathscr{U}$ 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 $\exists \hat{u} \in \mathscr{U}, \hat{u} \neq u$ such that all of these hold:

1. $r \in \bigcup_{s \in SU^{-1}(\hat{u})} SR(s)$
2. $\lambda_i \in \mu|_{\{t1\}}(u)$
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.

2.3.3. *Proximity constraints*

Using $\mathscr{M} = \{\mathscr{S}, \mathscr{T}, \mathscr{U}, \tau, \mu, \delta\}$ 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 Kirkpatrick et al. (2011), except we remove the assumption of geographical proximity and spatial roles. The simplified grammar for a proximity constraint clause can be written as:

```
C  ::= C ∨ C
     | C ∧ C
     | ¬C
     | S Q n role unit thr
S  ::= weak | strong
Q  ::= at most | at least | ϵ
```

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 $\hat{U} \subseteq \mathscr{U}$ such that

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

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

3. $\forall u \in \hat{U}\ \exists s \in SU^{-1}(u)$ such that $r \in SR(s)$
4. $\forall u \notin \hat{U} \nexists s \in SU^{-1}(u)$ 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 $\mathscr{C}$ denote the set of proximity constraints in this language.[5]

2.3.4. *Proximity-based RBAC model*

We can now conclude this section with our formal definition of a proximity-based RBAC model. Let $\mathscr{M} = \{\mathscr{S}, \mathscr{T}, \mathscr{U}, \tau, \mu, \delta\}$ denote a proximity model as defined previously. Policies would be based on **proximity tuples** $pt = \langle r, t, c \rangle$, where $c \in \mathscr{C}$ is a proximity constraint, $t \in$ *types* is a type associated with the requesting user's feature, and r is the requested role. Specifically, if $\mathscr{P}$ denotes the set of all such tuples, $\mathscr{A}$ denotes the set of actions, and $\mathscr{O}$ denotes the set of objects, a **proximity-based RBAC policy** would be the relation $Pol : \mathscr{P} \times \mathscr{A} \times \mathscr{O}$. 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 2 presents examples of policies for the five proximity realms.

3. Enforcement architecture

Designing a generic architecture that works across different applications and realms is crucial but challenging task. Different types of proximity and organizational settings have

[4] We note that this problem also arises in asynchronous deployments that work in different realms. However, the timing problem is heightened within the temporal realm.

[5] 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.

Table 2 – Example policies for various realms.

Geographical

Example: An officer is allowed to read a secret file only if no civilian is present with in 500 m and at least one senior officer is present in the same room.

$types = \{room, meters\}$, $\mathscr{O} = \{SecretFile\}$, $\mathscr{A} = \{read\}$,
$\mathscr{R} = \{SeniorOfficer, Officer, Civilian\}$

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 $pt = \langle Officer, room, C_1 \wedge C_2\rangle$

Policy: $\{pt, read, SecretFile\}$

Attribute

Example: A dating site member can view my profile if they have same profession and are no more than 10 years older.

$types = 2^{\{profession,\ age\}}$, $\mathscr{O} = \{MyProfile\}$, $\mathscr{A} = \{view\}$,
$\mathscr{R} = \{Member, Self\}$

Proximity Constraints $C_1 = \langle weak, \epsilon, 1, Self, \{profession\}, 0\rangle$,
$C_2 = \langle weak, \epsilon, 1, Self, \{age\}, 10\rangle$

Proximity tuple $pt = \langle Member, \{profession, age\}, C_1 \wedge C_2\rangle$

Policy: $\{pt, view, MyProfile\}$

Social

Example: A member of SACMAT network is allowed to view my conference album only if he is a friend of a friend or closer.

$types = \{individual, network, hops\}$, $\mathscr{O} = \{ConfAlbum\}$, $\mathscr{A} = \{view\}$,
$\mathscr{R} = \{Self, SACMATMember\}$

Proximity Constraints $C = \langle strong, \epsilon, 1, Self, hops, 2\rangle$

Proximity tuple $pt = \langle SACMATMember, individual, C\rangle$

Policy: $\{pt, view, ConfAlbum\}$

Cyber

Example: A manager can edit a shared Google document only if he is in a GoogleTalk session with a senior manager.

$types = \{GoogleTalk\}$, $\mathscr{O} = \{document_1\}$, $\mathscr{A} = \{write\}$,
$\mathscr{R} = \{Manager, SeniorManager\}$

Proximity Constraints $C = \langle weak, at\ least, 1, SeniorManager, GoogleTalk, 0\rangle$

Proximity tuple $pt = \langle Manager, GoogleTalk, C\rangle$

Policy: $\{pt, write, document_1\}$

Temporal

Example: A supervisor can only sign an employee's time card within 24 h after the employee did.

$types = \{hours, card\ signature\}$, $\mathscr{O} = \{time\ \ card\}$, $\mathscr{A} = \{sign\}$,
$\mathscr{R} = \{Employee, Supervisor\}$

Proximity Constraints $C = \langle weak, at\ least, 1, Employee, hours, 24\rangle$

Proximity tuple $pt = \langle Supervisor, card\ signature, C\rangle$

Policy: $\{pt, sign, time\ card\}$

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. 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. We propose a generic architecture and discuss changes that are required in it to accommodate different types of feature acquisition and communication strategies.

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*. Feature attestation could be accomplished using cryptographic techniques, such as digitally signed proofs of location, timestamps, or credentials. The central server serves as the Policy Information Point (PIP) (Oasis, 2004) and is responsible for evaluating the proximity constraints. The Policy Decision Point (PDP) uses the result of this constraint evaluation to facilitate the proper functioning of the Policy Enforcement Point (PEP). The main components of our architecture are as follows.

- *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 authorization server (see below) submits 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

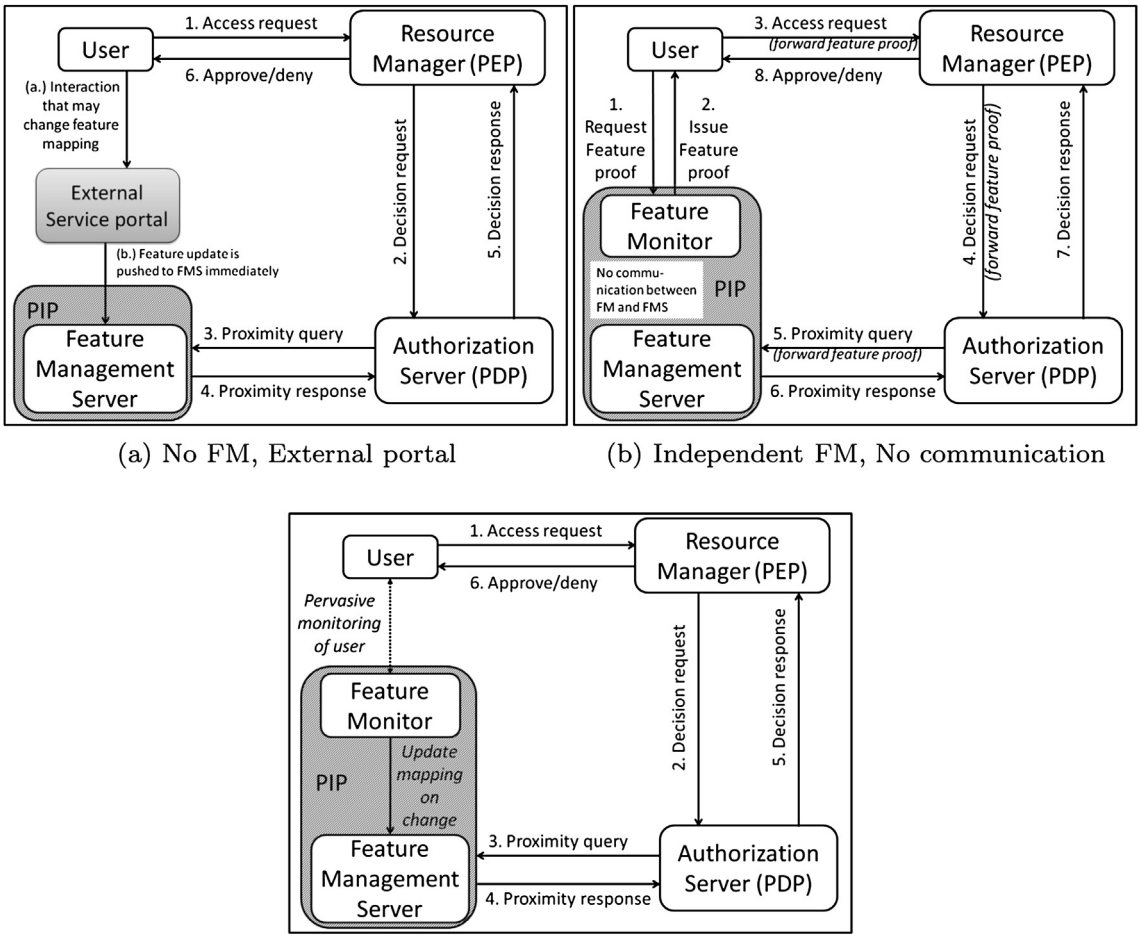

Fig. 1 – Enforcement architecture.

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 Fig. 1b. In an alternate architecture (Fig. 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 (Fig. 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 for feature attestation. 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 Fig. 1a.

3.1. *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 Fig. 1 and discussed below.

3.1.1. *No FM, external portal*

In this approach, illustrated in Fig. 1a, the user explicitly interacts with an external service portal (*e.g.*, a social network website 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). This feature update is immediately pushed by the external service portal to

the FMS so that the FMS can correctly evaluate proximity constraints.

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.

3.1.2. *Independent FM, no communication*

In this approach, users interact with a distributed set of entities (feature monitors) that have no direct communication links to the FMS. These feature monitors 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 Fig. 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.

3.1.3. *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 Fig. 1c). This approach is more appropriate for real-time geographical proximity where the location of user is pervasively tracked. This approach is also good for temporal proximity. For instance, the sensor may consist of a program that monitors updates to event log files and sends updates when the file changes.

3.2. *On the limitations of enforcement*

Deployment of such a proximity-based system would require addressing a couple of common technical hurdles through the use of heuristics. Specifically, designers of such systems need to understand that it is infeasible to automate policy analysis and absolute guarantees of correct enforcement cannot be guaranteed. The former claim derives from the fact that proximity constraints are Boolean expressions. If we let $\mathscr{D}$ denote the decision problem to determine whether or not an arbitrary proximity constraint can be satisfied, then the complexity of SAT (Cook, 1971) makes $\mathscr{D}$ NP-hard. Note that this is not saying that constraints cannot be evaluated efficiently (they can, in fact), just that it is intractable to determine if an arbitrary constraint can be satisfied.

The second challenge applies to asynchronous deployments, which would be the typical case, especially in the geographical realm. The same can be said for other realms if the users are distributed throughout the world. Evaluating a proximity constraint, then, turns into a consensus protocol. That is, the RM, AS, and FMS would need a consistent view of a dynamic world. Unless these principals have synchronous monitoring of the entire system, it has been proven that guaranteeing consensus is impossible (Fischer et al., 1985). Consequently, system designers must account for implementation details, such as network lag, updates to features, etc.

Given these constraints, we will proceed in the next section with *heuristic-based protocol templates*, which assume that practical implementations will account for such theoretical limits.

4. Heuristic-based protocol templates

Our aim in this section is to provide templates for enforcement protocols for architectures defined in Section 3. We have designed these protocols to support a number of enforcement goals, which we will formalize later. For now, our goals can be enumerated as

- validate users' claims for authorization to activate a role
- evaluate proximity constraints subject to a limited time frame
- minimize the amount of information leakage to prevent impersonation attacks
- prevent replay attacks by authorized users
- prevent improper accesses by unauthorized intruders

Our protocols employ standard cryptographic primitives. Specifically, let (*Gen*, *Enc*, *Dec*) denote an encryption scheme that provides indistinguishable encryptions under chosen plaintext attacks (IND-CPA-secure) such that $\mathrm{Gen}(1^n)$ denotes a probabilistic key generation algorithm with security parameter 1^n, $\mathrm{Enc}_k(\cdot)$ denotes encryption using the key k while $\mathrm{Dec}_k(\cdot)$ denotes the corresponding decryption. Next, let (*Gen*, *Sign*, *Ver*) denote a MAC scheme that is unforgeable against chosen message attacks (CMA-secure).[6]

We also adopt the standard convention that $\leftarrow$ denotes probabilistic assignment, while $:=$ denotes a deterministic assignment. Finally, while we use Enc and Dec generically, we distinguish between symmetric and public key encryption based on the key used. For instance, Enc_{K_p} refers to symmetric encryption using the key K_p for some identifier p. $\mathrm{Enc}_{sk(p)}$ denotes encryption using the secret key of p, while $\mathrm{Dec}_{pk(p)}$ would denote the corresponding decryption using p's public key.

In addition to standard cryptographic primitives, our protocols employ a number of additional building blocks, as follows. Recall that $\mathscr{U}$ denotes the set of users, $\mathscr{R}$ the set of roles, $\mathscr{O}$ the set of objects, $\mathscr{A}$ the set of actions, $\mathscr{S}$ the set of features in the reference space, $\mathscr{P}$ the set of proximity constraints, and *Pol* the set of policies. In addition, we adopt the convention that $\{0,1\}^n$ denotes a binary stream encoding some value (such as a cryptographic certificate). Lastly, as *Auth* (authentication primitive as described below) may take more than two

[6] For simplicity of notation, we use *Gen* to denote the key generation scheme for both encryption and MAC.

Protocol $\mathcal{Q}_0$ – base request protocol

0) **Initialization**

$[U]\ K_r \leftarrow \mathrm{Gen}(1^n)$
$[U]\ \sigma_{kr} \leftarrow \mathrm{Enc}_{pk(RM)}(K_r)$
$[U]\ \sigma_{ur} \leftarrow \mathrm{Enc}_{K_r}(obj, act, z, ts)$
$[U]\ \sigma_{ua} \leftarrow \mathrm{Enc}_{pk(AS)}(role, id_{user}, cred_{role}, cred_{feat}, z, ts)$

1) **Access request**:

$[U \rightarrow RM]\ \sigma_{ur}, \sigma_{ua}, \sigma_{kr}$

$[RM]\ K_r := \mathrm{Dec}_{sk(RM)}(\sigma_{kr})$
$[RM]\ (obj, act, z, ts) := \mathrm{Dec}_{K_r}(\sigma_{ur})$
$[RM]\ \gamma_r := \mathrm{Sign}_{sk(RM)}(obj, act, z, ts)$
$[RM]\ \sigma_{ra} \leftarrow \mathrm{Enc}_{pk(AS)}(\sigma_{ua}, obj, act, \gamma_r)$

2) **Decision request**:

$[RM \rightarrow AS]\ \sigma_{ra}$

$[AS]\ (\sigma_{ua}, obj, act, \gamma_r) := \mathrm{Dec}_{sk(AS)}(\sigma_{ra})$
$[AS]\ (role, id_{user}, cred_{role}, cred_{feat}, z, ts) := \mathrm{Dec}_{sk(AS)}(\sigma_{ua})$
$[AS]\ valid_{req} := \mathrm{Ver}_{pk(RM)}(\{obj, act, z, ts\}, \gamma_r)$
$[AS]\ auth_{id} := \mathrm{Auth}(id_{user}, \{cred_{role}, role\})$
$[AS]\ \langle pol_1, \ldots, pol_m \rangle := \mathrm{FindPolicies}(role, obj, act)$
$[AS]\ \gamma_{af} := \mathrm{Sign}_{sk(AS)}(id_{user}, \langle pol_1.pt, \ldots, pol_m.pt \rangle, z)$
$[AS]\ \sigma_{af} \leftarrow \mathrm{Enc}_{pk(FMS)}(id_{user}, \langle pol_1.pt, \ldots, pol_m.pt \rangle, cred_{feat}, z, \gamma_{af}, ts)$

3) **Proximity query**:

$[AS \rightarrow FMS]\ \sigma_{af}$

$[FMS]\ (id_{user}, \langle pol_1.pt, \ldots, pol_m.pt \rangle, cred_{feat}, z, \gamma_{af}, ts) := \mathrm{Dec}_{sk(FMS)}(\sigma_{af})$
$[FMS]\ valid_{pol} := \mathrm{Ver}_{pk(AS)}(\{id_{user}, \langle pol_1.pt, \ldots, pol_m.pt \rangle, z\}, \gamma_{af})$
$[FMS]\ auth_{feat} := \mathrm{Auth}(id_{user}, cred_{feat})$
$[FMS]\ \langle res_1, \ldots, res_m \rangle := \mathrm{EvalTuples}(\langle pol_1.pt, \ldots, pol_m.pt \rangle)$
$[FMS]\ \gamma_f := \mathrm{Sign}_{sk(FMS)}(\langle res_1, \ldots, res_m \rangle, id_{user}, z)$

4) **Proximity response**:

$[FMS \rightarrow AS]\ \langle res_1, \ldots, res_m \rangle, \gamma_f$

$[AS]\ valid := \mathrm{Ver}_{pk(FMS)}(\{\langle res_1, \ldots, res_m \rangle, id_{user}, z\}, \gamma_f)$
$[AS]\ res := \mathrm{Decide}(\langle pol_1[res_1/pol_1.pt], \ldots, pol_m[res_m/pol_m.pt] \rangle)$
$[AS]\ \gamma_a := \mathrm{Sign}_{sk(AS)}(res, obj, act, z)$

5) **Decision response**:

$[AS \rightarrow RM]\ res, \gamma_a$

$[RM]\ valid_{res} := \mathrm{Ver}_{pk(AS)}(\{res, obj, act, z\}, \gamma_a)$
$[RM]\ retval := act[obj]$
$[RM]\ \sigma_{res} \leftarrow \mathrm{Enc}_{K_r}(retval)$

6) **Approve or deny**:

$[RM \rightarrow U]\ \sigma_{res}$

Fig. 2 – Protocol for architecture in Fig. 1a.

parameters (the first is always a user, each additional is a binary-encoded value), we use $(\{0,1\}^n)^+$ to denote the function takes one or more binary parameters.

- *Auth* : $\mathscr{U} \times (\{0,1\}^n)^+ \rightarrow \{true, false\}$ – a non-interactive authentication scheme that takes a user ID and one or more binary credentials as input, returning *true* or *false* to indicate whether the authentication succeeds
- *FindPolicies* : $\mathscr{R} \times \mathscr{O} \times \mathscr{A} \rightarrow 2^{Pol}$ – identifies the relevant policies for the requested role, object, action tuple
- *EvalTuples* : $(\mathbb{N} \rightarrow 2^{\mathscr{P}}) \rightarrow (\mathbb{N} \rightarrow \{true, false\})$ – evaluates a sequence of proximity tuples for the current feature mapping μ, returning a sequence of truth values declaring whether or not the associated tuple was satisfied[7]
- *Decide* : $2^{Pol} \rightarrow \{true, false\}$ – determines which policies were satisfied and returns a Boolean indicating whether or not to grant access
- *Bind* : $\mathscr{U} \times \mathscr{R} \times \mathscr{S} \rightarrow \{0,1\}^n$ – a computationally binding procedure that produces a verifiable credential (*e.g.*, a

[7] Observe that a sequence can be modeled as a partial function from the naturals to a set of items to be sorted. *E.g.*, if s is a sequence, $s(1)$ denotes the first item, $s(2)$ the second, etc.

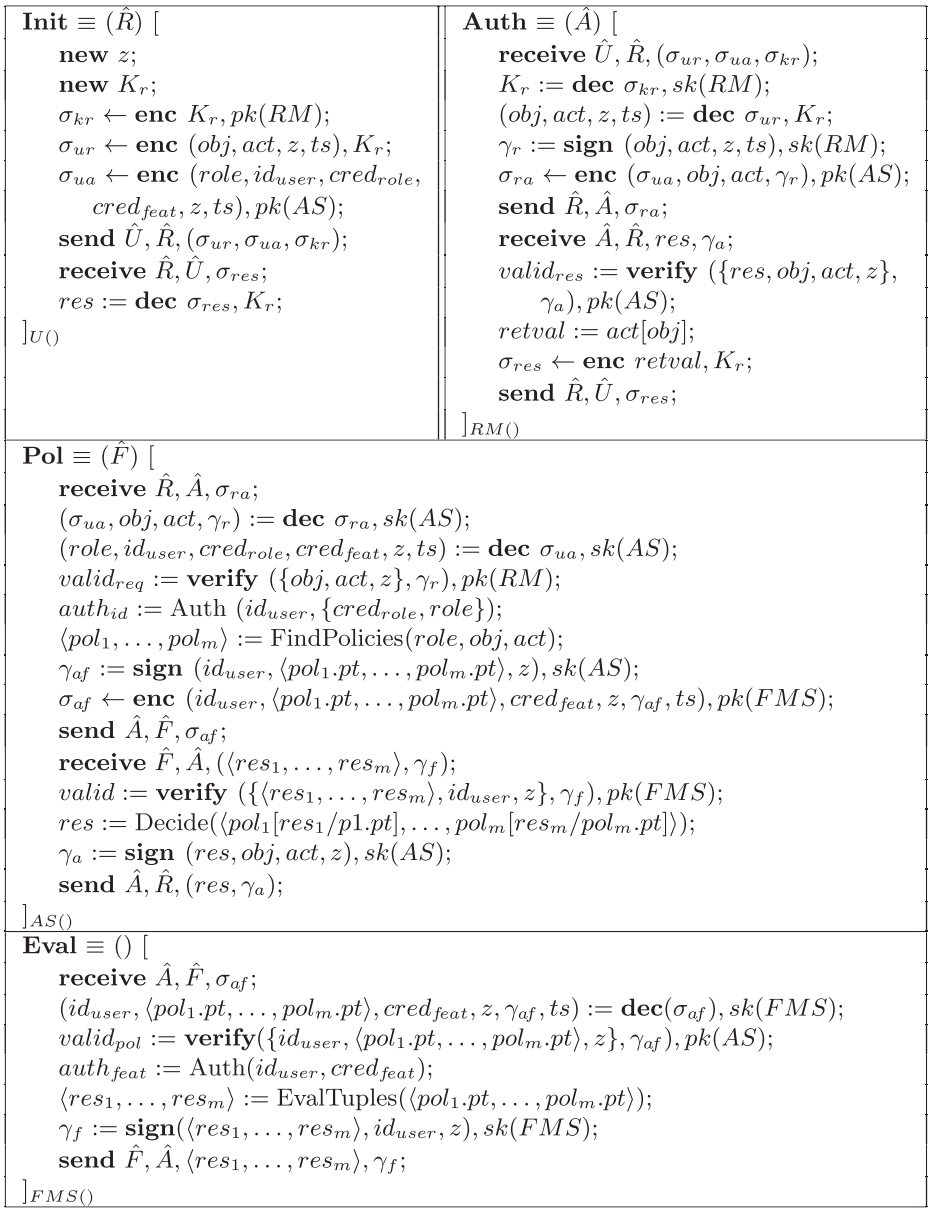

Fig. 3 – PCL specification for Protocol $\mathscr{C}_0$.

digitally signed certificate) that ties the user to the requested role and the claimed feature at the time requested

- *GenValidation*: $\{0,1\}^n \rightarrow (\{0,1\}^n \rightarrow \{true, false\})$ – takes a digital credential as input and returns a function that can be applied to validate the credential at a later time

Protocol $\mathscr{C}_0$, as shown in Fig. 2, describes the data exchanged for Fig. 1a. In this architecture, the external service portal pushes updates to FMS as needed. As this portal is considered external to our architecture, communication with it is not modeled in Protocol $\mathscr{C}_0$. Instead, Protocol $\mathscr{C}_0$ shows the data exchanged when the request is made. We use the notation $pol_i.pt$ denotes the proximity tuple *pt* for the given policy pol_i (See Section 2.3.4). Observe that *Decide* does not declare when the decision should be made to grant access, as this is application specific. That is, some systems may require all policies to be satisfied, while others grant access if any policy is satisfied.

Protocol $\mathscr{C}_0$ introduces several variables that may warrant additional clarification. To start, $obj \in \mathscr{O}$ denotes the object under consideration, $act \in \mathscr{A}$ is the requested action, z is a nonce, and *ts* denotes a timestamp. We use $cred_{feat}$ and $cred_{role}$ to denote credentials that attest to one's authorization to use a feature or activate a role.

To begin to analyze the security qualities of Protocol $\mathscr{C}_0$, we can formalize the protocol using PCL (Datta et al., 2007), as shown in Fig. 3. In PCL notation, the protocol is re-structured from the perspective of various *roles*[8] that specify the

[8] This is an unfortunate collision of terminology. The term "role" in relation to PCL should not be confused with the notion of RBAC role.

$\theta_U = \{obj, act, role, id_{user}, cred_{role}, cred_{feat}, ts\}$
$\theta_{RM} = \emptyset$
$\theta_{AS} = \{Auth, FindPolicies\}$
$\theta_{FMS} = \{Auth, (\mathcal{U} \times \mathcal{S})\}$
$\phi_{U,R} = \{retval, z, K_r\}$
$\phi_{RM,R} = \{K_r, obj, act, z, res, retval, ts\}$
$\phi_{AS,R} = \{obj, act, role, id_{user}, cred_{role}, cred_{feat}, z, ts, auth_{id}, \langle res_1, \ldots, res_m \rangle\}$
$\phi_{FMS,R} = \{id_{user}, \langle pol_1.pt, \ldots pol_m.pt \rangle, cred_{feat}, z, ts\}$
$\phi_{\mathcal{A},R} = \{\langle res_1, \ldots res_m \rangle, res\}$

Fig. 4 – Knowledge gained during execution R of Protocol $\mathscr{C}_0$.

behavior of various participants within the protocol. That is, instead of looking at the global view of the protocol, each participant's actions are viewed in isolation. In Protocol $\mathscr{C}_0$, for instance, **Init** designates the initiator role. In an honest execution, the user *U* can take on the role of initiator, which requires knowledge of the resource manager *RM* in charge of the protected resource. Similarly, **Auth** is the authorization role, **Pol** is the policy management role, and **Eval** is the proximity evaluation role.

Also, note that there is a distinction between the participant of the protocol (*e.g.*, $\widehat{R}$) and the associated principal (*e.g.*, *RM*). This distinction is important, as the participant may be an adversary attempting an attack on the system. That is, $\widehat{R}$ may actually be the adversary $\mathscr{A}$. As such, the PCL specification makes this distinction obvious.

The advantage of this formalization is that it makes explicit what data is seen by each participant in the protocol, and we can infer what knowledge is gained during an execution *R* of the protocol. Fig. 4 shows the knowledge gained by each participant during execution. In this notation, θ_i denotes the *a priori* knowledge of principal *i*, while $\phi_{i,R}$ denotes the knowledge gained from execution *R*. We use $\phi_{\mathscr{A},R}$ to denote the knowledge gained by a probabilistic polynomial-time (PPT) adversary with only access to the public keys of the participants. We also write $\phi_i \vDash \tau$ to indicate that principal *i* *has* or *knows* the piece of information τ. Note that $\phi_i \vDash \tau$ implies $\tau \in \phi_i \cup \theta_i$ or τ can be derived from some $\hat{\tau} \in \phi_i \cup \theta_i$. For simplicity, we omit any encrypted message σ_j from the $\phi_{i,R}$ sets, as our assumptions regarding encryption presume that the knowledge gain from just an encrypted message is negligible. We also omit verifications of MACs, as these are only relevant to determining the origin integrity of a message, rather than providing true information about the data exchanged.

As a final note before presenting our security analysis, our analysis focuses on a specific adversarial model. Specifically, we assume the Dolev–Yao adversarial model (Dolev and Yao, 1983), in which an adversary can eavesdrop or modify any message. Furthermore, our analysis focuses on *rational* attacks. That is, we assume that *RM*, *AS*, and *FMS*, participate honestly unless they could benefit from deviating. In fact, as these principals have a vested interest in protecting the resource, we find no rational attacks by them, with the exception of violating the desired privacy guarantees. As such, our analysis assumes honest participation by these principals, except where noted. Instead, we focus on attacks in which an authorized user attempts to exceed his or her privileges (*e.g.*, eavesdropping on another user reading a file), or external adversaries attempting to gain illicit access to the system. In either case, the adversary would benefit by deviating, so we find these attacks rational and focus on them in our analysis. Lastly, for Protocol $\mathscr{C}_0$, we exclude the feature portal from our analysis and consider it to be a trusted third party.

Lemma 1. Replay attacks by an external adversary are detectable except with negligible probability.

Proof. Assume that *z* is nonce that is *n* bits in length. Then the probability that two users select the same *z* in two separate runs of the protocol is $1/2^n$. Furthermore, σ_{ua} also depends on the timestamp *ts*. If we let *m* denote the number of protocol runs that can be initiated within $ts \pm \delta$, where δ denotes the maximum time for which the timestamp *ts* would be valid, then the probability that two randomly selected nonces are the same would be $m/2^n$. For moderate values of *n*, this probability is negligible. Thus, by keeping a log of the most recent *m* nonces used, RM would detect the replay with near certain probability.

Furthermore, even if *RM* fails to detect the replay, the only valid strategy for an adversary $\mathscr{A}$ would be to replay the exact messages. That is, as $\phi_{\mathcal{A},R} \nvDash id_{user}, cred_{role}, cred_{feat}$, $\mathscr{A}$ cannot forge a message $\widehat{\sigma_{ua}}$ for which $auth_{id}$ would be successful, other than the original σ_{ua}. In such a case, the effect of the replay would be contingent upon act. If *act* involves a modification, the replay would simply repeat the modification. Again, though, this succeeds with only negligible probability. However, if *act* is a read, the attack cannot succeed at all, as $\phi_{\mathcal{A},R} \nvDash K_r$. Thus, $\mathscr{A}$ cannot decrypt the object and the attack fails. □

Lemma 2. Tampering by an external adversary fails except with negligible probability.

Proof. This property follows from the fact that (*Gen*, *Enc*, *Dec*) is IND-CPA-secure and (*Gen*, *Sign*, *Ver*) is CMA-secure. As such, $\mathscr{A}$ cannot forge $\widehat{\sigma_i}$ for any of the encrypted messages σ_i or $\widehat{\gamma_j}$ for any of the MACs γ_j. Thus, any attempt at tampering with the messages would be detected by the honest recipient. At best, $\mathscr{A}$ could induce a denial-of-service by modifying, for instance, any of the res_i values, causing γ_f to fail verification. However, the adversary cannot forge any message that would be accepted as legitimate, except with negligible probability. □

Protocol $\mathcal{Q}_1$ – feature monitor with no direct communication

1) $[U \rightarrow FM]\ id_{user}, role$
   $[FM]\ cred_{feat} := \mathrm{Bind}(id_{user}, role, feat)$
   $[FM]\ valid_{feat} := \mathrm{GenValidation}(cred_{feat})$
2) $[FM \rightarrow U]\ cred_{feat}, valid_{feat}, \widehat{ts}$
   $[U]\ \sigma_{uf} \leftarrow \mathrm{Enc}_{pk(FMS)}(valid_{feat}, \widehat{ts})$
3) $[U \rightarrow RM]\ \sigma_{ur}, \sigma_{ua}, \sigma_{kr}, \sigma_{uf}$
4) $[RM \rightarrow AS]\ \sigma_{ra}, \sigma_{uf}$
5) $[AS \rightarrow FMS]\ \sigma_{af}, \sigma_{uf}$
   $[FMS]\ (valid_{feat}, \widehat{ts}) := \mathrm{Dec}_{sk(SMF)}(\sigma_{uf})$
6) $[FMS \rightarrow AS]\ \langle res_1, \ldots, res_m \rangle, \gamma_f$
7) $[AS \rightarrow RM]\ res, \gamma_a$
8) $[RM \rightarrow U]\ \sigma_{res}$

Fig. 5 – Protocol for architecture in Fig. 1b.

Theorem 1. Protocol $\mathscr{C}_0$ is secure under the Dolev–Yao adversarial model.

Proof. Follows from the preceding two lemmas. □

Lemma 3. Protocol $\mathscr{C}_0$ preserves user privacy from the RM.

Proof. This follows from the fact that $\phi_{RM} \not\models id_{user}$. That is, RM is able to receive an authenticated evaluation of the proximity constraints without having to know the identity of the requesting user. □

Lemma 4. Protocol $\mathscr{C}_0$ prevents replay by authorized users.

Proof. The primary concern here is that a user may exploit the asynchronous, distributed nature of the protocol to exercise a right when the proximity constraint no longer holds. However, $\phi_{RM,R} \models ts$, ensuring RM has the ability to validate that the timestamp claimed is reasonably accurate. Furthermore $\phi_{AS,R} \models ts$ and $\phi_{FMS,R} \models ts$, ensuring all principals see the same timestamp. Finally, γ_r is derived from ts, which allows AS and (indirectly) FMS to validate that the timestamp matched the time of request, as checked by RM. Thus, the timestamp and the MACs ensure that messages exchanged match the time used to evaluate the proximity constraint. □

Theorem 2. Protocol $\mathscr{C}_0$ provides strong authentication of user credentials and proximity claims.

Proof. Note that a successful execution R of $\mathscr{C}_0$ requires $\phi_{AS,R} \models auth_{id}$. At the same time, $\phi_{AS,R} \models auth_{id} \supset \theta_U \models cred_{role}$. Thus, strong user authentication is provided by the assumptions regarding the *Auth* primitive. Additionally, $\phi_{FMS,R} \models auth_{feat} \supset \theta_U \models cred_{feat}$. As this credential is created by the centralized portal, which is beyond the scope of attack by a PPT adversary $\mathscr{A}$, the protocol integrates strong proximity authentication. □

Protocol $\mathscr{C}_0$ is appropriate for proximity constraints defined for a centralized application. That is, social proximity assumes the use of a social network application with a global view. Similarly, cyber proximity is generally built on the assumption of a centralized service, such as telephony, though peer-to-peer designs (*e.g.,* Skype) also exist; in the latter case, Protocol $\mathscr{C}_0$ would be inappropriate. Temporal proximity can be ensured assuming actions can be synchronized within the system. Applying Protocol $\mathscr{C}_0$ for geographical proximity would be very challenging, as pervasive location monitoring is difficult.

Finally, note that there is a possibility for performance optimizations in certain deployments of Protocol $\mathscr{C}_0$. Specifically, if AS and FMS are hosted on the same machine, the

$\theta_U = \{obj, act, role, id_{user}, cred_{role}, ts\}$
$\theta_{RM} = \emptyset$
$\theta_{AS} = \{Auth, FindPolicies\}$
$\theta_{FMS} = \{Auth, (\mathcal{U} \times \mathcal{S})\}$
$\phi_{U,R} = \{retval, z, K_r, cred_{feat}, valid_{feat}, \widehat{ts}\}$
$\phi_{RM,R} = \{K_r, obj, act, z, res, retval, ts\}$
$\phi_{AS,R} = \{obj, act, role, id_{user}, cred_{role}, cred_{feat}, z, ts, auth_{id}, \langle res_1, \ldots, res_m \rangle\}$
$\phi_{FMS,R} = \{id_{user}, \langle pol_1.pt, \ldots pol_m.pt \rangle, cred_{feat}, valid_{feat}, \widehat{ts}, z, ts\}$
$\phi_{\mathcal{A},R} = \{\langle res_1, \ldots res_m \rangle, res, cred_{feat}, valid_{feat}, \widehat{ts}\}$

Fig. 6 – Knowledge gained during execution R of Protocol $\mathscr{C}_1$.

encryption of σ_{af} and the MAC γ_f are extraneous. That is, if the data exchanged by these two principals occurs over a secure channel in which eavesdropping is not possible, then the additional cryptographic protections are not necessary.

Fig. 5 shows Protocol $\mathscr{C}_1$, which extends the previous protocol to facilitate communication for the architecture in Fig. 1b. Most of the protocol is identical, with the exception being the generation of $cred_{feat}$. In Protocol $\mathscr{C}_0$, this credential is assumed to be generated by the portal and is tangential to the protocol. In Protocol $\mathscr{C}_1$, however, the user must explicitly retrieve the credential, which binds the user to a role and feature at a given timestamp $\widehat{ts}$. In this scenario, *Bind* is assumed to be a computationally binding, perfectly hiding commitment scheme, while *GenValidation* is a non-interactive proof. For instance, the two primitives may constitute a zero-knowledge proof-of-knowledge. The key is that *U* must not be able to forge such a credential in polynomial time. The remainder of the protocol is identical, with the exception that the proof needs to be forwarded to *FMS*, which verifies the credential.

The analysis of Protocol $\mathscr{C}_1$ is virtually identical to Protocol $\mathscr{C}_0$. Fig. 6 shows the PCL knowledge sets for Protocol $\mathscr{C}_1$. In this scenario, we are assuming the simplest case, in which the exchange between *U* and *FM* occurs in an insecure manner. For instance, this data may be exchanged over unencrypted Wi-Fi. Formally, this means $\phi_{\mathscr{A},R} cred_{feat}, valid_{feat}, \widehat{ts}$. However, the following lemmas show that this is not a security threat. Alternatively, this point could be made moot by using a secure channel between *U* and *FM*.

Lemma 5. Protocol $\mathscr{C}_1$ remains secure against a PPT adversary under the Dolev–Yao model.

Proof. As shown in Fig. 6, $\phi_{\mathscr{A},R} \vDash cred_{feat}, valid_{feat}, \widehat{ts}$. However, as *Bind* is assumed to be perfectly hiding, $\phi_{A,R} \nvDash cred_{role}$. Furthermore, as *Bind* is computationally binding, $\mathscr{A}$ could not forge the credential within polynomial time, even with the knowledge in $\phi_{\mathscr{A},R}$. Thus, $\mathscr{A}$ cannot forge σ_{ua} with the stolen credential in a manner that would be accepted by *AS*, except with negligible probability as described previously. Therefore, the stolen credential cannot be used to create unauthorized access. □

Lemma 6. Protocol $\mathscr{C}_1$ retains the privacy protection against *RM* as Protocol $\mathscr{C}_0$.

Proof. This follows from the fact that $\phi_{RM,A}$ is identical for the two protocols. Thus, Protocol $\mathscr{C}_1$ continues to protect user privacy. □

Lemma 7. Protocol $\mathscr{C}_1$ continues to provide strong feature authentication for authorized users.

Proof. The computationally binding nature of Bind prevents forgery of $cred_{feat}$ by *U* except with negligible probability. Thus, *FMS*, when validating the credential, has probabilistic assurance that the credential has not been forged. Furthermore, as $\phi_{FMS,R} \vDash \widehat{ts}, ts$, *FMS* can determine that the request was made within an acceptable time frame of the feature credential creation. □

Finally, Fig. 7 shows Protocol $\mathscr{C}_2$, which defines an extension for the architecture in Fig. 1c. As in $\mathscr{C}_1$, *FM* is responsible for issuing a computationally binding credential. However, the binding routine in this protocol is more flexible. That is, in Protocol $\mathscr{C}_1$, $valid_{feat}$ must be implemented as a non-interactive technique, such as a cryptographically signed certificate. In contrast, in Protocol $\mathscr{C}_2$, *FM* pushes the credential validation information to *FMS*. For instance, *FM* could generate a Pedersen commitment (Pedersen, 1992), sending the commitment to *FMS* while sending the data to open the commitment to the user. Note, though, that this protocol includes the same knowledge sets as $\mathscr{C}_1$, with the exception that $\phi_{U,R} \nvDash valid_{feat}$. As such, the security analysis remains unchanged.

5. Related work

Role based access control (RBAC) (Sandhu et al., 1996) is a permission model that grants access based on roles that users have as a part of an organization. Several extensions to RBAC have been proposed that attempt to incorporate various contextual factors while making access decisions. GEO-RBAC (Damiani et al., 2007) and LRBAC (Ray et al., 2006) consider

Protocol $\mathcal{Q}_2$ – feature monitor protocol

1) $[U \rightarrow FM]\ id_{user}, role$
   $[FM]\ cred_{feat} := \text{Bind}(user, role, feat)$
   $[FM]\ valid_{feat} := \text{GenValidation}(cred_{feat})$
2) $[FM \rightarrow FMS]\ valid_{feat}$
3) $[FM \rightarrow U]\ cred_{feat}$
4) $[U \rightarrow RM]\ \sigma_{ur}, \sigma_{ua}, \sigma_{kr}$
5) $[RM \rightarrow AS]\ \sigma_{ra}$
6) $[AS \rightarrow FMS]\ \sigma_{af}$
7) $[FMS \rightarrow AS]\ \langle res_1, \ldots, res_m \rangle, \gamma_f$
8) $[AS \rightarrow RM]\ res, \gamma_a$
9) $[RM \rightarrow U]\ \sigma_{res}$

Fig. 7 – Protocol for architecture in Fig. 1c.

location of the user requesting access while Gal and Atluri (2000) consider temporal attributes such as time of access as a factor in decision making. STARBAC (Aich et al., 2007), Lot RBAC (Chandran and Joshi, 2005) and Atluri and Chun (2007) incorporate both location of the user and time of access into the access control model. While these consider some specific contextual factors, Covington et al. (2001), Kulkarni and Tripathi (2008) and Zhang and Parashar (2004) take a more general approach by designing access control framework that can incorporate a variety of contextual factors. Our work incorporates the proximity to other users in various realms as a factor in access control decisions. SRBAC (Hansen and Oleschuk, 2003) and Kirkpatrick and Bertino (2010) consider spatial and temporal constraints for mobile RBAC systems while our approach is applicable to a more general domain.

Prox-RBAC (Kirkpatrick et al., 2011) extended the notion of spatially aware RBAC to consider the relative locations of other users within an indoor space model (Jensen et al., 2010; Jensen et al., 2009), 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 (Georgiadis et al., 2001) incorporates contextual information into team-based access control by actively monitoring ongoing interactions. PBAC (Didar-Al-Alam et al., 2010; Gupta et al., 2006) 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.

6. 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 three generic enforcement architectures and provided protocol templates for enforcing such systems, formalized these protocols using PCL and proved security properties of these protocols. In summary, we argue that it is feasible to deploy a practical proximity-based RBAC system for a variety of contextual factors.

REFERENCES

Aich S, Sural S, Majumdar AK. STARBAC: spatiotemporal role based access control. In: Proceedings of the 2007 OTM confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS – volume part II, OTM'07. Berlin, Heidelberg: Springer-Verlag; 2007. p. 1567–82.

Atluri V, Chun SA. A geotemporal role-based authorisation system. Int J Inform Comput Secur 2007;1:143–68.

Chandran S, Joshi J. LoT RBAC: a location and time-based RBAC model. In: Proc. of 6th international conference on web information systems engineering (WISE). Springer-Verlag; 2005. p. 361–75.

Clementini E, Felice PD, Oosterom Pv. A small set of formal topological relationships suitable for end-user interaction. In: Proc. of 3rd international symposium on advances in spatial databases (SSD). London, UK: Springer-Verlag; 1993. p. 277–95. URL, http://dl.acm.org/citation.cfm?id=647223.718898.

Cook SA. The complexity of theorem-proving procedures. In: Proc. of 3rd annual ACM symposium on theory of computing, STOC'71. New York, NY, USA: ACM; 1971. p. 151–8.

Covington MJ, Long W, Srinivasan S, Dev AK, Ahamad M, Abowd GD. Securing context-aware applications using environment roles. In: Proc. of 6th ACM symposium on access control models and technologies (SACMAT'01) 2001. p. 10–20.

Damiani ML, Bertino E, Catania B, Perlasca P. GEO-RBAC: a spatially aware RBAC. ACM Trans Inform Syst Secur 2007;10(1).

Datta A, Derek A, Mitchell JC, Roy A. Protocol composition logic (PCL). Electron Notes Theor Comput Sci 2007;172:311–58.

Didar-Al-Alam SM, Mahmud H, Mottalib MA. Modifications in proximity based access control for multiple user support. Int J Eng Sci Tech 2010;2:3603–13.

Dolev D, Yao A. On the security of public-key protocols. IEEE Trans Inform Theor 1983;2.

Fischer MJ, Lynch NA, Paterson MS. Impossibility of distributed consensus with one faulty process. J ACM 1985;32:374–82.

Gal A, Atluri V. An authorization model for temporal data. In: Proceedings of the 7th ACM conference on computer and communications security, CCS '00. New York, NY, USA: ACM; 2000. p. 144–53.

Georgiadis CK, Mavridis I, Pangalos G, Thomas RK. Flexible team-based access control using contexts. In: Proc. of 6th ACM symposium on access control models and technologies (SACMAT). New York, NY, USA: ACM; 2001. p. 21–7.

Gupta SKS, Mukherjee T, Venkatasubramanian K, Taylor TB. Proximity based access control in smart-emergency departments. In: Proc. of 4th Annual IEEE international conference on pervasive computing and communications workshops (PERCOMW). Washington, DC, USA: IEEE Computer Society; 2006. p. 512.

Hansen F, Oleschuk V. SRBAC: a spatial role-based access control model for mobile systems. In: Proc. of 8th Nordic workshop on secure it systems (NORDSEC) 2003. p. 129–41.

Jensen CS, Lu H, Yang B. Graph model based indoor tracking. In: 10th International conference on mobile data management (MDM) 2009. p. 122–31.

Jensen CS, Lu H, Yang B. Indoor – a new data management frontier. IEEE Data Eng Bull 2010;33(2):12–7.

Kirkpatrick MS, Bertino E. Enforcing spatial constraints for mobile RBAC systems. In: Proc. of 15th ACM symposium on access control models and technologies (SACMAT). New York, NY, USA: ACM; 2010. p. 99–108.

Kirkpatrick MS, Damiani ML, Bertino E. Prox-RBAC: a proximity-based spatially aware RBAC. In: Proc. of 19th ACM SIGSPATIAL international conference on advances in geographic information systems (GIS) 2011. p. 339–48.

Kulkarni D, Tripathi A. Context-aware role-based access control in pervasive computing systems. In: Proceedings of the 13th ACM symposium on access control models and technologies, SACMAT '08. New York, NY, USA: ACM; 2008. p. 113–22.

Oasis. Oasis extensible access control markup language (XACML) TC. Spring; 2004. p. 1–16. 2009 (May 5) URL, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

Pedersen TP. Non-interactive and information-theoretic secure verifiable secret sharing. In: Proc. of 11th Annual international

conference on advances in cryptology, CRYPTO '91. London, UK: Springer-Verlag; 1992. p. 129–40.
Ray I, Kumar M, Yu L. LRBAC: a location-aware role-based access control model. In: Proc. of International conference on information systems security (ICISS) 2006. p. 147–61.
Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. IEEE Comput 1996;29(2):38–47.
Zhang G, Parashar M. Context-aware dynamic access control for pervasive applications. In: Proc. communication networks and distributed systems modeling and simulation conference 2004. p. 21–30.

Aditi Gupta is currently pursuing her PhD in the Department of Computer Science at Purdue University under the supervision of Dr. Elisa Bertino. Her research interests include access control, security in context aware systems, malware defense and building secure architectures. She can be contacted at: aditi@purdue.edu.

Dr. Michael S. Kirkpatrick is an assistant professor of computer science at James Madison University. His main research interests focus on security engineering, access control, applied cryptography, privacy, and secure operating systems. Prior to earning graduate degrees from Purdue University and Michigan State University, he spent several years working in the field of semiconductor optical proximity correction for IBM Microelectronics (later Server & Technology Group) in Essex Junction, VT. He can be contacted at kirkpams@jmu.edu.

Dr. Elisa Bertino is professor of computer science at Purdue University, and serves as Director of Purdue Cyber Center and Research Director of CERIAS. Prior to joining Purdue, she was a professor and department head at the Department of Computer Science and Communication of the University of Milan. She has been a visiting researcher at the IBM Research Laboratory (now Almaden) in San Jose, at the Microelectronics and Computer Technology Corporation, at Rutgers University, at Telcordia Technologies. Her recent research focuses on database security, digital identity management, policy systems, and security for web services. She can be contacted at bertino@purdue.edu.